

Multimedia Information Retrieval

by

Mark David Dunlop B.Sc.(Hons.)

Being a thesis submitted for the degree of Doctor of Philosophy in the
Department of Computing Science at the University Of Glasgow.

7 October 1991

© Mark D. Dunlop 1991

Table of Contents

Table of Contents	i
Table of Figures	iv
Acknowledgements	1
Declaration of Originality.	2
Permission to Copy	2
Summary	3
1 Introduction	5
1.1 Thesis Aims	5
1.2 Motivation	5
1.2.1 Multimedia Information	5
1.2.2 Multimedia Libraries	6
1.3 Thesis Outline	7
2 Review of Information Retrieval	10
2.1 Introduction	10
2.2 Free Text Retrieval	11
2.2.1 General Description	12
2.2.2 Calculating Descriptors	12
2.2.3 Query Matching	16
2.2.4 Relevance Feedback	20
2.2.5 User Interfaces	22
2.2.6 Evaluating IR Systems	25
2.2.7 Multimedia Access	28
2.2.8 Large Scale Usage	28
2.3 Hypermedia	30
2.3.1 General Description	32
2.3.2 Link Types	32
2.3.3 Node Types	34
2.3.4 Alternative Models of Hypermedia	34
2.3.5 Conversion From Traditional Texts	35
2.3.6 Navigational Issues	36
2.3.7 Evaluating HM Systems	36
2.3.8 Multimedia Access	40
2.3.9 Large Scale Usage	40
2.4 Combining Free Text Retrieval and Browsing	41
2.4.1 General Description	41
2.4.2 Search and Query in a Hypermedia Network	42
2.4.3 Virtual Structures for Changing Information	44
2.4.4 Access to Hybrid Document Bases	46

2.5 MultiMedia Document Systems	47
2.6 Context of Thesis	49
2.7 Summary	50
3 A Model of Access to Non-Textual Nodes	51
3.1 Introduction	51
3.2 Simple Approach	52
3.3 Simple Cluster Approach	52
3.4 Extended Cluster Approaches	53
3.4.1 Level 1 Cut Off	54
3.4.2 Level 2 Cut Off	54
3.5 Consideration of Type Information	56
3.6 Mixed Descriptor Types	57
3.6.1 A Multi-Descriptor Example	58
3.6.2 Querying	59
3.6.3 Relevance Feedback	60
3.6.4 Suitable Descriptors	60
3.7 Limitations of Model	60
3.8 Conclusions	61
3.9 Full Definitions for Sample Document Base	61
3.9.1 Simple Approach	62
3.9.2 Basic Cluster Approach	62
3.9.3 Extended Cluster Approaches	63
3.9.4 Mixed Descriptor Types	66
4 Justification of Model	68
4.1 Introduction	68
4.2 Test Collection	68
4.3 Experimental Procedure	69
4.3.1 Calculating Index Descriptors	70
4.3.2 Calculating Cluster Descriptors	70
4.3.3 Comparing Descriptors	71
4.4 Results	71
4.5 Limitations of Experiment	73
4.6 Conclusions	74
5 Relevance Feedback	75
5.1 Introduction	75
5.2 Vector Based Justification	76
5.3 Experimental Justification	78
5.3.1 Equipment	78
5.3.2 Process	79
5.3.3 Results	80

5.4 Negative Feedback	81
5.5 Probabilistic Model	84
5.6 Conclusions	84
6 mmIR – A Prototype Implementation	86
6.1 Introduction	86
6.2 Document Base	86
6.3 Access	88
6.3.1 Query-Only Access	88
6.3.2 Browsing-Only Access	90
6.3.3 Hybrid Access	92
6.4 Internal Design	94
6.4.1 Indexing Textual Nodes	94
6.4.2 Image Indexing Algorithm	95
6.4.3 Query Processing	96
6.4.4 Relevance Feedback	97
6.4.5 Nearest Neighbour Calculations	98
6.4.6 History Mechanism	99
6.4.7 Data Acquisition	100
6.4.8 Software Development	101
6.5 Suggested Improvements	101
6.6 Conclusions	103
7 User Testing	104
7.1 Introduction	104
7.2 Experiment Overview	104
7.3 Questionnaires	105
7.3.1 Prior Experience Questionnaire	105
7.3.2 Test Questionnaire	105
7.3.3 Performance Questionnaire	106
7.3.4 Understanding Check	107
7.4 Sessions	107
7.4.1 User Experience Results	107
7.4.2 User Logs	109
7.4.3 Test Equipment	110
7.5 Analysis of Logs	110
7.6 Analysis Results	112
7.6.1 Time Taken	112
7.6.2 Quality of Solutions	118
7.6.3 Nodes Accessed	119
7.6.4 User Confidence	121
7.6.5 User Speed Expectations	124

7.6.6 User's Surprise at Results	126
7.6.7 Query Length	127
7.6.8 History Usage	128
7.6.9 Expert Trial	128
7.7 Observation Results	131
7.7.1 Relevance Feedback	132
7.7.2 Disorientation	133
7.7.3 Query Style	133
7.7.4 Link Types	134
7.8 Suggested Improvements	134
7.8.1 Integration of Queries and Browsing	134
7.8.2 Presentation of Matching Score	136
7.8.3 Provision of Maps	138
7.8.4 Interface Style	138
7.9 Limitations of Tests	139
7.10 Conclusions	139
7.11 Sample Questionnaires	140
8 Conclusions	155
9 Bibliography	159

Table of Figures

Figure 2.1	Average Precision-Recall Curve	26
Figure 2.2	A Sample Petri-Net	35
Figure 2.3	Example hypermedia network.	37
Figure 2.4	Electrical equivalent of example hypermedia network	37
Figure 3.1	Sample document base	51
Figure 3.1	Sample document base	62
Figure 4.1	Distribution of CACM record size	69
Figure 4.2	Distribution of number of links per CACM record	69
Figure 4.3	Cosine correlation between cluster and index based descriptors (by link count)	72
Figure 4.4	Cosine correlation between cluster and index based descriptors (by descriptor size)	73
Figure 5.1	Diagrammatic addition of query vector to feedback vector	76
Figure 5.2	Distribution of matching coefficient scores	79
Figure 5.3	Effect of positive feedback on CACM queries	80
Figure 5.4	Effect of positive feedback for various values of k	81

Figure 5.5	Diagrammatic subtraction of query vector from feedback vector	82
Figure 5.6	Effect of negative feedback on CACM queries	82
Figure 5.7	Effect of negative feedback for various values of k	83
Figure 6.1	A typical mmIR screen	86
Figure 6.2	Distribution of Links	87
Figure 6.3	Distribution of node size in the Highway Code	88
Figure 6.4	Query window	88
Figure 6.5	Node window for query access	89
Figure 6.6	Navigator window for query access	89
Figure 6.7	Query menus	90
Figure 6.8	Node window for home node	91
Figure 6.9	Node window for browsing access	91
Figure 6.10	Navigator window for browsing access	91
Figure 6.11	Node window for hybrid access	93
Figure 6.12	Navigator window for hybrid access	93
Figure 6.13	Browsing in the hybrid model	94
Figure 6.14	Example use of double stack history	100
Figure 7.1	Graph of prior knowledge of the Highway Code	108
Figure 7.2	Graph of computer usage	108
Figure 7.3	Graph of Macintosh experience	109
Figure 7.4	Time taken by users to answer each question on their questionnaire	112
Figure 7.5	Individual time graphs	113
Figure 7.6	General results for time taken per question	115
Figure 7.7	Average time between questions	116
Figure 7.8	Time taken per question plus mean understanding time	116
Figure 7.9	General results for time taken per question	117
Figure 7.10	Average score which users solutions achieved.	118
Figure 7.11	Overview of users' scores	119
Figure 7.12	Users' scores against time taken	119
Figure 7.13	Number of nodes accessed by users in order to answer each question	120
Figure 7.14	Graph of users' expectation of answer correctness	121
Figure 7.15	Graph of user confidence for non-answered solutions	122
Figure 7.16	Graph of nodes accessed after visiting answer node	123
Figure 7.17	Average number of nodes accessed after visiting answer node	124
Figure 7.18	Graph of how many steps users felt they took to answer the questions.	124

Figure 7.19	Graph of how users rated the computer's response time.	125
Figure 7.20	Graph of how many users felt they could do better with a computer based Highway Code	126
Figure 7.21	Graph of user's surprise at their results	126
Figure 7.22	Length of first query for each question (in words)	127
Figure 7.23	Uses of history commands per user	128
Figure 7.24	Total time taken by expert user.	129
Figure 7.25	Mean total time taken by expert user.	129
Figure 7.26	Number of nodes accessed by expert user.	130
Figure 7.27	Mean number of nodes accessed by expert user.	130
Figure 7.28	Difference between number of nodes accessed by expert and test users	131
Figure 7.29	Length of expert user's queries	131
Figure 7.30	Sample network from prototype application	135
Figure 7.31	Sample network with explicit matched node list	136
Figure 7.32	Graph of score against position in matched node list	137
Figure 7.33	Example retrieval node with score graph	137

Acknowledgements

Although this thesis presents my individual work there are many people who contributed to it by their discussion and support. Firstly I thank Keith van Rijsbergen, my first supervisor, who's discussion, criticism, and guidance have been invaluable throughout my studentship. I also extend my gratitude to my second supervisor, Phil Gray, for his help throughout my research. Thanks to Isobel Baxter from the Department of Psychology for help in developing the questionnaires and, with Paddy O'Donnell and Steve Draper, for helping me prepare the user tests. Thanks also to Duncan Sinclair and Tunde Cockshott for initial run throughs of the user tests, and for the thirty *guinea-pigs*. David England and Mark Sanderson are gratefully thanked for taking the time and effort to read over and suggest improvements to this thesis.

The Computing Science Department and, in particular, the joint psychology Glasgow Interactive Systems cenTre (GIST) have provided an excellent environment in which to study and research. Thanks also to Pete Bailey for his help in providing equipment for my rather heavy experiments, and to the entire support, technical, and secretarial staff of the computing department who have been very helpful throughout my study.

Finally, thanks to my family, friends, and especially to Julia for all there help, encouragement, and support.

Funding for the research reported here was provided by the Science and Engineering Research Council (SERC) through student award reference 88802857.

Declaration of Originality.

The material presented in this thesis is entirely the result of my own independent research carried out in the Department of Computing Science at the University of Glasgow, under the supervision of Prof. C.J. van Rijsbergen. Any published or un-published material that is used by me has been given full acknowledgment in the text.

Permission to Copy

Permission to copy without fee all or part of this thesis is granted provided that the copies are not made or distributed for direct commercial advantage, and that the name of the author, the title of the thesis, and its date of submission are clearly visible on the copy.

Summary

With recent advances in screen and mass storage technology, together with the on-going advances in computer power, many users of personal computers and low end work-stations are now regularly manipulating non-textual information. This information may be in the form of drawings, graphs, animations, sound, or video (for example). With the increased usage of these media on computer systems there has not, however, been much work in the provision of access methods to non-textual computer based information.

An increasingly common method for accessing large document bases of textual information is free text retrieval. In such systems users typically enter natural language queries. These are then matched against the textual documents in the system. It is often possible for the user to re-formulate a query by providing relevance feedback, this usually takes the form of the user informing the system that certain documents are indeed relevant to the current search. This information, together with the original query, is then used by the retrieval engine to provide an improved list of matched documents. Although free text retrieval provides reasonably effective access to large document bases it does not provide easy access to non-textual information. Various query based access methods to non-textual document bases are presented, but these are all restricted to specific domains and cannot be used in mixed media systems.

Hypermedia¹, on the other hand, is an access method for document bases which is based on the user browsing through the document base rather than issuing queries. A set of interconnected paths are constructed through the base which the user may follow. Although providing poorer access to large document bases the browsing approach does provide very natural access to non-textual information. The recent explosion in hypermedia systems and discussion has been partly due to the requirement for access to mixed media document bases.

Some work is reported which presents an integration of free text retrieval based queries with hypermedia. This provides a solution to the scaling problem of browsing based systems, these systems provide access to textual nodes by query or by browsing. Non-textual nodes are, however, still only accessible by browsing – either from the starting point of the document base or from a textual document which matched the query.

A model of retrieval for non-textual documents is developed, this model is based on document's context within the hypermedia document base, as opposed to the document's content. If a non-textual document is connected to several textual documents, by paths in the hypermedia, then it is likely that the non-textual document will match the query whenever a high enough proportion of the textual documents match. This model of retrieval uses clustering techniques to calculate a descriptor for non-textual nodes so that they may be retrieved directly in response to a query. To establish that this model of

¹ also hypertext

retrieval for non-textual documents is worthwhile an experiment was run which used the text only CACM collection. Each record within the collection was initially treated as if it were non-textual and had a cluster based description calculated based on citations, this cluster based descriptor was then compared with the actual descriptor (calculated from the record's content) to establish how accurate the cluster descriptor was. As a base case the experiment was repeated using randomly created links, as opposed to citations. The results showed that for citation based links the cluster based descriptions had a mean correlating of 0.230 with the content based description (on a range from 0 to 1, where 1 represents a perfect match) and performed approximately six times better than when random links were used (mean random correlation was 0.037). This shows that citation based cluster descriptions of documents are significantly closer to the actual descriptions than random based links, and although the correlation is quite low, the cluster approach provides a useful technique for describing documents.

The model of retrieval presented for non-textual documents relies upon a hypermedia structure existing in the document base, since the model cannot work if the documents are not linked together. A user interface to a document base which gives access to a retrieval engine and to hypermedia links can be based around three main categories:

- browsing only access, use the retrieval engine to support link creation
- query only access, use links to provide access to non-text
- query and browsing access

Although the last user interface may initially appear most suitable for a document base which can support queries and browsing it is also potentially the most complex interface, and may require a more complex model of retrieval for users to successfully search the document base. A set of user tests were carried out to establish user behaviour and to consider interface issues concerning easy access to documents which are held on such document bases. These tests showed that, overall, no access method was clearly better or poorer than any other method. The traditional view that hypermedia was easier to use by novices, but free text querying was better for experts, was supported to a certain extent but the differences were not large. The tests also raised several areas for consideration when building a user interface to a document base with some hypermedia structure and a retrieval engine.

The provision of query and browsing access within a user interface also raises an issue concerning relevance feedback: in a traditional retrieval engine the user could only give relevance feedback on documents which matched the last query (since these are the only documents which can be accessed). In a system which allows the user to browse the neighbourhood of matched document it is possible that the user will view, and thus give relevance feedback, on documents which do not match the query. A discussion and experiments are presented which show that the effect of feedback follows intuition for positive feedback, but that negative feedback, under the vector space model, is not as intuitive and cannot be considered an inverse operation to positive feedback.

1 Introduction

...once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, “and what is the use of a book,” thought Alice, “without pictures or conversations”?

Lewis Carroll: Alice’s Adventures In Wonderland

1.1 Thesis Aims

The research reported in this thesis was conducted with two main aims. Firstly, to provide a general method for query-based access to non-textual documents. Since, in general, the content of a non-textual document cannot be used directly for retrieval, the model was to be based on an indirect approach. To provide a suitable environment for this approach, document bases which were composed of interconnected documents were chosen as the underlying storage structure. The second aim of the thesis was derived from the choice of underlying document base. There are very few systems which provide browsing facilities, through interconnected documents, and full query facilities. Consequently, many implications of this form of document base have not been considered. The research reported here aimed to analyse the main implications from a user interface viewpoint.

Overall, the research planned to develop a model of access to document bases, which support browsing and querying, which is natural, effective, and suitable for mixed-media document bases.

1.2 Motivation

The motivation behind this thesis comes from two directions. Firstly, the requirement to provide access to non-textual documents held in large computer based document bases. Secondly, by considering the current methods for accessing such document bases. This section provides an overview of the motivation and also serves as an introduction to general terminology used throughout the thesis.

1.2.1 Multimedia Information

Computers have traditionally been used for processing numerical and textual information, the vast majority of computers are now used almost exclusively for processing textual documents. There are, however, many fields of work which require access to non-textual information; for example medics require access to x-rays, architects to building plans, ornithologists to bird calls, and estate agents to property photographs, to name but a few. In many of these fields the non-textual information is at least as important as the textual information which may accompany it: it is hard to conceive a building company being given the plans for an office block in prose. In other areas the non-textual information is used to highlight details or to give alternative views of lengthy textual documents. There are also occasions when an essentially textual document must be viewed as an image, for

example it is useful to see the entire layout of a newspaper page and not just the text composing the stories, insurance companies must also keep a photographic copy of claim forms so that any comments in margins or corrections can be seen as well as the actual text of the form.

With recent advances, in quality and price, of display and storage technology, computers are being used more regularly for the production of images, animation, and music. It is no longer the case that just because a document is held on computer it is forced not to contain any pictures or conversations. It is becoming apparent to many computer users that it is possible to create large libraries of documents which contain mixed textual and non-textual information.

1.2.2 Multimedia Libraries

Most existing non-textual libraries are held on non-computer media. Local libraries often have extensive music libraries held on audio cassette, vinyl, or compact disc, and video stores have ranges of movies on video tape or disc. These libraries are either indexed by author, artist, title, or by a rough classification – these are textual identifiers which are used to describe the non-textual medium. Libraries traditionally accessed textual documents by the same process, for example novels are indexed typically by author and title. In recent years there has been a significant growth in computer-based library systems which have access to the entire text of the document (or at least a paragraph or two extracted from the document). This not only allows the searcher to partially examine the content of the document, say an academic paper, without going to the shelf and retrieving it, but allows searches to be based on the content of the document – not just on restricted features such as authors and title. In a textual environment reasonably effective general purpose algorithms have been developed which allow the user to input a natural language sentence which is then matched against all the documents in the electronic library. Documents are then retrieved which match the user's sentence (or query), and are hopefully relevant to the current requirement. No such general purpose algorithms exist, or are likely to exist, for the automatic matching of non-textual documents by query. The problem of accessing non-textual nodes by query has been solved by either associating a piece of text with each non-textual document, this text is then used for matching, or with domain specific solutions. The use of a textual entry representing a non-textual document, though at first promising, is likely to lead to many problems since these descriptions must be created by an indexer. This is not only a time consuming task, but is also likely to be unreliable since human indexers may produce inconsistent and biased descriptions due to their own perspective and level of domain knowledge, for example an auctioneer would index the Mona Lisa very differently from a fine art student.

Partly to provide access to multimedia document bases (electronic libraries), there has been a rapid increase in the popularity of browsing-based hypermedia systems in recent years. These systems allow the user to browse through the document base using a series

of paths connecting documents together. These paths need not be restricted to accessing textual nodes and often access nodes in many media, providing a very natural environment for the storage and retrieval of non-textual information. Browsing-based retrieval systems are, however, restricted in scale due to the undirected approach users must take. The conflict between browsing and querying may be compared with a book library. When using a small library, e.g. a small department library, or when looking within a field which one is very familiar with, it is often easier to simply browse through the bookcases looking for books which are useful. The organisation of the library and any labels which are provided will help locate the required books. Alternatively, when looking for books in an unknown domain in a large library it is much easier to start with a query, either to the librarian for help or to the catalogue system (whether computerised or not). This conflict has led to the inclusion of query routines in hypermedia systems, so that the user can issue a query to locate the approximate areas to browse. These queries cannot, however, provide direct access to non-textual nodes, leaving users to browse to these from textual nodes.

In systems which provide access to the document base by query and by browsing, any non-textual documents must be linked on paths from textual documents, otherwise it would not be possible to access them. These links can be used to provide access to non-textual nodes directly by query, based on the non-textual document's context in the document base; for example if a digitised image of the Mona Lisa were linked with various textual nodes (e.g. art essays and auctioneers reports) then if a reasonable proportion of these textual nodes were relevant to the user's query, it is likely that the image itself will be relevant and should also be considered as a possible match. To make use of this form of access to non-textual information a combined hypermedia and free text retrieval model of information retrieval must be used.

A combined model of information retrieval which encompasses elements from both free text retrieval (querying) and hypermedia (browsing), may lead to a more complex model of retrieval, which may in turn lead to reduced user performance. User testing of an information retrieval system which can be accessed either through browsing, querying, or by both provides a test environment in which to establish user patterns and therefore, hopefully, predict elements of an interface which would make for easier access to the information.

1.3 Thesis Outline

This thesis considers the provision of access to non-textual information and related issues. Chapter 2 provides a description of work which has been carried out in the areas of free text (query based) retrieval, hypermedia browsing, their combination, and work in the direct retrieval of non-textual documents by query. Chapter 2 also presents formal definitions of each access method. These definitions, though not describing every feature of the access methods, describe the essential features of each access method.

A model of retrieval for non-textual documents is developed in chapter 3. This model is based on an underlying document base in which users can access information by browsing and by querying. The model itself is based on using context information from the hypermedia structure to provide access, by query, to nodes whose content cannot be used for retrieval. The chapter develops the model to various levels of complexity, ranging from the current approach, not retrieving non-textual nodes by query, to a general recursive definition which can be expanded as required. Each level is described in a variation of set notation similar to that used in chapter 2.

An experiment to justify the model developed in chapter 3 as providing reasonably effective retrieval is presented in chapter 4. This experiment is based on a test only document base and compares the description of documents calculated from their content (the document's index) with that calculated as if the document were non-textual (i.e. using the techniques described in chapter 3).

Chapter 5 presents a discussion on the effects of a combined browsing and querying environment on relevance feedback. Relevance feedback is a significant element of free text retrieval systems by which users can mark certain documents as being relevant to the current query. This information, together with the last query, can then be used to provide an improved list of matched documents. Relevance feedback can, traditionally, only be given on documents which match a query, since these are the only documents a user can view. Within an environment in which the user can also browse, (s)he can access documents which do not match the query. Consequently, relevance feedback can be given on non-matching documents. A model of this effect is developed, and tested experimentally, to show the varying effect of feedback depending on whether the document matched the original query (and the strength of that matching).

A prototype system, *mmIR*, is presented in chapter 6. *mmIR* provides access to the British Highway Code by hypermedia browsing, free text queries, or a combination of both. It was mainly developed to show that the model presented in chapter 3, for accessing non-textual nodes by content, was valid and for use in user tests. This chapter presents a discussion of the system from a user's viewpoint: describing use of the system, for each access method, and the functionality of each command. It also presents an overview of the major internal design features of the systems, and of algorithms used.

The user tests themselves are reported in chapter 7. These tests were mainly carried out to establish whether any of the access methods (browsing, querying, or a combination) provides more effective access to a document base such as the Highway Code. The tests were carried out on 30 users who had reasonable knowledge of computer usage, but had no knowledge of computing science and could not be described as expert users. The results of the tests are described quantitatively, mainly in graph form, and qualitatively.

Finally, a summary of the work presented in this thesis and conclusions which can be derived from it is given in chapter 8.

2 Review of Information Retrieval

2.1 Introduction

Information retrieval is a very general phrase which can be applied to much of the work which computers perform during an average day, and much of the research in computing science. Morrissey, Harper, and van Rijsbergen (1986) described eight forms of query which a typical office may have to handle:

“DBMS¹ type. Here we specify that certain parts of an office object must contain specific values. An example is ‘give me all memos written by Smith and sent to Jones during September’.

DRS² type. Topicality is the important feature here. We are interested in office objects that are about a certain subject. For example, we might be interested in all memos and letters concerning office holidays.

DRS/DBMS mixture. These are a combination of A and B [DBMS and DRS] above, such as retrieve all letters written by J. Bloggs and sent to J. Smith about office holidays.

Computational queries. For these queries, which require a specific answer, information must be aggregated, such as how many people work in personnel? Who earns the most money in personnel? What was the total amount paid out in expenses for the personnel department last year?

Fact queries. These require that information be extracted from office objects, such as in a legal office one might ask ‘what is the maximum penalty for armed robbery?’.

System-data queries. Examples are: What’s in the system? What can I browse through? What office objects are there? What do they look like? Show me what the employee records look like.

Help queries. Where am I? What can I do now? What did I do last? Can I see an example of that operation?

Command queries. We interpret ‘query’ in a very general way. It includes not only retrieval queries to perform the operations described above. For example, queries to modify stored data.”

This thesis concerns itself entirely with searching in document retrieval systems, that is searches in which the user is looking for a document, or a set of documents, which are on a specific topic of interest. This form of retrieval is very general and can be used, in part, to answer many other forms of search, e.g. factual and help queries could both be answered by documents retrieved by a topical search. There are two methods of searching for

¹ DataBase Management System

² Document Retrieval System

documents on a particular topic which are very different in their approach: *browsing*, which provides the user facilities to explore a structured document base, and *querying*, which provides the user with a facility to retrieve documents in response to a query. In the context of document retrieval these two approaches are represented by hypermedia browsing and free text querying (or information retrieval).

This chapter presents an introduction and review of the work which has been carried out in document retrieval, it concentrates on work which is relevant to retrieval system user interfaces or the access to non-textual documents, but a general overview of the field is also given. This chapter is split into four main sections which discuss work on querying (or free text retrieval), browsing (or hypermedia/hypertext), combining queries with browsing, and work on multimedia document bases. These sections introduce the basic technologies, terminology, and notation which are used throughout the thesis. Each access method is described formally, using set notation, as are many commonly used algorithms. The formal definitions of each access method define the basic structure and facilities of that method. Although very simple, these models do define the essence of the access methods. Each section also discusses various extensions to the basic model, which include many of the facilities and enhancements which are found in working systems. This chapter ends with short sections placing the remainder of the thesis in context and summarising this chapter.

2.2 Free Text Retrieval

Free text retrieval (also known as document retrieval, DR, or simply as information retrieval, IR) permits access to documents by content. These documents may be held on very large document bases. For example Stein (1991) quotes the Library of Congress which has a collection of roughly 25 terabytes (25×10^{12} bytes). In information retrieval systems, users initiate a search by entering a query, the system replies by searching the document base for all documents which match the given query, and then by presenting these to the user. The query is typically in the form of a boolean algebra statement or, with increasing predominance, a phrase in natural language (e.g. English, French, or Gaelic).

This section describes various aspects of free text retrieval systems: the general construction of the document base, how documents are described for the purpose of retrieval, how queries are processed, and how users can provide feedback to the retrieval engine. The section then discusses various aspects of user interfaces to such systems. Throughout the section the terms *recall* and *precision* are used to refer to, respectively, the fraction of documents which are relevant to the current query,³ which are actually retrieved, and the fraction of those documents retrieved which are actually relevant. These terms are widely used in the evaluation of retrieval systems and are discussed in more detail at the end of this section.

³ that are held in the document base

2.2.1 General Description

Document bases for information retrieval systems can be considered as a set of independent documents, each document being composed of a *content* and a *descriptor*. The content of a document is the text (or other media) which the user is presented with when the document matches a given query. Traditionally this content was a small piece of text which was representative of the document (e.g. the abstract of an academic paper) but more recent systems, known as full text retrieval systems, have stored the entire document on line. Since the content of documents may be quite large, a descriptor is usually calculated for each document which more concisely describes its content; this descriptor is then used for matching against queries rather than the document's content. Using a slight variation of set notation the document base can be described as follows. Throughout this thesis double-square brackets, e.g. $\llbracket C \rrbracket$, are used to denote the *evaluation* of a document's content to produce a descriptor, this evaluation can be performed by simple indexing or more complex processing.

$\text{Base} = \{ N_i \mid i \in \mathbb{D} \}$	<i>a set of all accessible documents</i>
where	
$\mathbb{D} \subset \mathbb{Z}^+$	<i>subset of positive integers (currently indexed document identifiers)</i>
$N_i = \langle C_i, D_i \rangle$	<i>a pair of content and descriptor</i>
$D_i = \llbracket C_i \rrbracket$	<i>the indexing/evaluation of content C_i</i>
C_i	<i>the content of the document</i>

The above definition provides a very general description of a free text retrieval document base. All that remains to be specified is the process by which descriptors are calculated from the documents' content, the structure of these descriptors, and the algorithm which is used to match queries against document descriptors. These processes are described in the next two sections.

To guarantee access to all documents in the document base it must be shown that all descriptors are non-empty, i.e. $\forall i \in \mathbb{D}: D_i \neq \emptyset$, where \emptyset is the empty descriptor. As discussed later in this chapter and in chapter 3, this requirement imposes restrictions on the media which can be accessed by this approach (e.g. it is very difficult to imagine a method for generating a descriptor for a raster image).

2.2.2 Calculating Descriptors

A descriptor of a document attempts to maintain information about the document which is likely to assist the retrieval engine in deciding whether the document matches any future queries. The descriptors for documents in information retrieval systems vary greatly in their complexity and this in turn relates to the complexity of the matching algorithm. If the system is to provide a complex matching algorithm then the descriptors must be structured

to support this algorithm. However, if the matching algorithm is simple then the descriptors should be as compact as possible (while providing enough information), to minimise the overheads (in time and space) of the retrieval engine.

Many models of information retrieval consider the document base to be made up of multi-dimensional vectors with each dimension referring to one of the variables in the retrieval system (e.g. a term or a phrase). The simplest descriptors model documents by an N-dimensional binary vector, where N is the number of distinct *terms* in the document base. A term can initially be defined as a word, resulting in a model which estimates the content of a document by tabulating the existence of words within that document. Each document can be considered as a set of terms, and expressed as follows:

$$D_i = \{ t \mid t \in T \wedge \text{occurs_in}(t, C_i) \}$$

where

T = the set of all terms used in the document base

C_i = the content of document i

occurs_in = text searching function which ensures that the term t occurs in the body of the document.

Although simplistic in approach this model does provide reasonably good retrieval performance, the effectiveness can, however, be considerably improved by use of two widespread techniques: *stop word filters* and *conflation algorithms*, these can be considered as extensions to the functionality of *occurs_in*. A list of words which are extracted from a document will contain many that have no meaning when taken out of context (e.g. *the*, *a*, *an*, *there*, and *some*), a reduction in the size of descriptors and an increase in retrieval effectiveness can be achieved by passing all words through a stop words filter, Van Rijsbergen (1979 pp. 18-19) provides a sample list of words which should be considered as stop words in most systems. It is, however, useful to include any words which occur in almost all documents in the stop words list, as these have very little power to discriminate relevant from non-relevant documents. The use of a stop words filter should decrease the total number of matches to a query and increase the precision of the retrieval system, since the remaining list of matched documents will contain a higher fraction of relevant documents. It will also significantly speed up retrievals because significantly fewer words will need to be analysed.

A significant improvement in retrieval performance can also be achieved by taking account of word inflections. This can be achieved by conflating each word before it is entered in the list of terms used to describe the document (for example *work*, *worked*, *working*, and *works* could be conflated to the single term *work*). Conflation results in a much reduced number of terms and a significant improvement in the recall of a retrieval system, since more documents contain each conflated term than contain each of the original words. A common conflation algorithm was developed by Porter (1980) and does

not rely upon a dictionary of inflections. The effectiveness of his algorithm does, initially, appear to be rather poor since many common words conflate to the same term when they should not (e.g. *communism* and *community*). The algorithm also does not result in actual words being formed (e.g. *communism* conflates to *commun*), which poses problems when a user interface displays the terms which it is using to search on (e.g. to allow the user to directly manipulate their weights). Porter, however, describes the results of tests which showed that the retrieval performance of a system using this algorithm was no poorer than the same system using a more complex, dictionary based, conflation algorithm.

The binary vector representation has been extended in some systems which use integer vector descriptors. These, model not only the existence of terms (or words) within a document but also, model the number of times that terms occur. This extension is based on the premiss that terms which are used more often within a document are more descriptive of that document than less frequently used terms. Some systems model this information as a vector of real numbers, such that each term is associated with a weight between zero and one which expresses how descriptive the term is. One common method for extending this information is the inverse document frequency (I.D.F.) algorithm developed by Sparck Jones (1972) which, given the number of documents a term occurs in, n , and the total number of documents, N , assigns a weight of $\log(N/n)$ to the given term⁴. The I.D.F. calculation increases the importance of terms which are least frequently used in the document base thus, hopefully, increasing the precision of the retrieval engine since this model of descriptors represents the document's meaning more accurately. The *log* function is required to prevent very infrequently occurring terms being given disproportionately high weights. At the extremes of term frequency the benefits of I.D.F. are easily observed: when a term occurs in all documents ($n=N$) the term is completely ignored and is given a weight of $\log(1)=0$, this is reasonable since the term should be included in the stop word list (as it is incapable of distinguishing relevant from non-relevant documents). At the other extreme when the term only occurs in one document ($n=1$) then the I.D.F. weight will be $\log(N)$ which equals 1.0 when $N=10$ and 4.0 when $N=10000$, again this is reasonable since if the user enters this term the retrieval system could be considered as badly failing if it did not match the only document to contain the term. Due to the computational complexity of calculating the inverse document frequencies (or any similar weighting scheme) these techniques are usually applied when constructing document descriptors. They can, however, be applied during the matching process, resulting in a document base which can be more easily updated but has a considerably slower matching algorithm. The I.D.F. algorithm does not take account of how descriptive a term is for each document. Since many retrieval systems record how often a term is used within a document, this information could be used together with

⁴ Sparck Jones's paper defines the weight given to a specific term as $f(N) - f(n) + 1$ where $f(x) = y$ such that $2^{y-1} < x \leq 2^y$. This can be approximated to be $\log(N/n) + 1$, the addition of 1 simply provides a term weight in the range 1 to infinity, the addition has been removed here so that a term which occurs in every document has a weight of 0.

I.D.F. to provide a more accurate description of how descriptive each term is for each node. An example of such a definition was developed as part of the experiment described in section 4.3.1. This defines the weight of a index term for a given document as:

$$W_{Di} = \frac{D_i}{D} \log \left(\frac{T}{T_i} \right)$$

where

W_{Di} = weight of term i in descriptor D

T = total number of term occurrences used in the collection

T_i = total number of occurrences of term i in the collection

D = total number of term occurrences in D

D_i = total number of occurrences of term i in D

The definition of descriptors as term-based is a simplification of modern free text retrieval. Many recent models of information retrieval are based on greater understanding of the natural language which they are processing. A common extension is to use noun phrases rather than individual words to create terms, this results in better matching since the descriptors are more powerful and should describe the document's content more closely. There is, however, a significant processing cost involved with using phrases for retrieval rather than index terms. Smeaton and Sheridan (1991) present work on the syntactic analysis of natural language with the specific aim of providing improved access to free text document bases. Their approach, which builds tree structures from the query and matches these with documents, has been mainly developed for English, but Finnish and Swedish morphological analysers have also been developed. They are currently conducting experiments to establish whether the approach can be scaled up to very large document bases. Many of the basic techniques from natural language processing were used in the French Yellow Pages (Clemencin 1988) in which the retrieval engine is used to access a business telephone directory. Example queries include (English translations of typical French queries are given here): “my wipers are broken”, “I am looking for a cross country skiing training course”, and “I live in Lannion and would like to rent a car”.

Traditional term based retrieval systems can be improved considerably in quality if word-senses are used as index terms instead of words. This approach would prevent one sense of the word from being retrieved upon a request to other senses of the word (for example, this would differentiate river **banks** from financial **banks**). This would increase the precision of the retrieval engine since only documents using the correct sense of a word will be retrieved. The disambiguation can either be performed as part of a natural language processor or can be performed statistically by analysing the words in the neighbourhood of the ambiguous word. Krovetz and Croft (1989) and Zernick (1991) discuss the use of machine readable dictionaries in word sense disambiguation.

2.2.3 Query Matching

After the document base has been indexed it can be accessed by query, traditionally these queries were constructed using boolean logic. Such queries express a logical combination of words which should appear in matched documents, often systems only provide the user with the **and**, **or**, and **not** operators. Example queries on a transport document base could be: ‘car **and** rail’, ‘(car **and** rail) **or** aeroplane’, and ‘aeroplane **and not** (British **and** Airways)’. This form of querying can be used to produce very precise queries. However, it is very difficult to produce complex queries and almost impossible for casual users to create all but the simplest of queries. Experiments by Verhoeff, Goffman, and Belzer (1961) showed that using traditional boolean queries was significantly less than optimal when considering a group of users, each with their own understanding of relevance, issuing the same query. They summarise their result by saying that “if the system responds to a request asking for ‘*a*’ **and** ‘*b*’ by giving the intersection of the responses it would have given to requests for ‘*a*’ and ‘*b*’ separately, it risks giving too much irrelevant material. If in response to a query for ‘*a*’ **or** ‘*b*’ it gives the union of the responses to the request for ‘*a*’ and ‘*b*’ if made separately, it risks leaving out relevant material.”⁵ Later, as an introduction to their work on an extended model of boolean retrieval, Salton, Fox, and Wu (1983) described four areas in which boolean retrieval performance is poor:

- The size of the matched document list is difficult to control: retrieval lists often fluctuate between far too many and far too few (e.g. zero) documents to be of use.
- The matched document list is not ordered, thus the user needs to examine the entire list rather than the top few most likely matches.
- Varying weights of terms in queries and in documents cannot be controlled, for example, a term which is just excluded from the stop word list is given exactly the same treatment as a term which only occurs once in the document base.
- Boolean queries provide no form of *fuzzy* matching, for example a series of terms which are **anded** together must **all** be present for a document to be retrieved, even if 99% of the terms are present the document will be considered as irrelevant as a document which contains none of the terms.

These restrictions can be partly overcome by expanding the query incrementally and presenting the results as a ranked list of documents, this partly simulates free text retrieval in a boolean environment. As an example consider the query $A \wedge B \wedge C$, the results of this query could be presented at the top of a list followed by the results of the following queries (excluding any nodes which have already been matched): $A \wedge B$, $B \wedge C$, $A \wedge C$, and finally $A \vee B \vee C$. While this approach can be successful, very long queries are formed when the query is more complex and it still does not solve the basic problem that users require to understand boolean logic before they can make a successful query – in

⁵ It would appear that Verhoeff *et al.* have interchanged the effects of using **and** and **or**. Using intersection to model **and** would be expected to leave out relevant material, while using intersection for **or** would risk giving too much irrelevant material.

everyday conversation the terms **and** and **or** are often freely exchangeable and the term **not** is very poorly understood when used in conjunction with other operators. This approach, of ranking boolean query results, can be combined with automatic boolean query generation. This provides an environment in which the user enters keywords or natural language (and possibly some other details), the system then queries a retrieval system using boolean algebra, and finally ranks the results and displays them to the user. Although this approach provides access to retrieval engines which can only accept boolean queries, it should not be considered as a general model of retrieval. Bovey and Robertson (1984) present an early description of how this can be achieved, while an edition of Information Processing and Management contains three papers on the subject: Salton (1988), Heine (1988), and Fox and Koll (1988).

More recent systems based on boolean queries have attempted to make use of form based user interfaces to simplify the process of creating queries (for example the Textriever system developed by Burkowski, 1991) but these typically reduce the power of boolean access. Many boolean retrieval systems have been extended to give greater control of queries, for example, the provision of proximity constructs to limit two terms to occurring in the same sentence, or within a few words of each other, in the original text. These lead to more precise, but considerably harder to formulate, queries.

A much simpler approach, from the user perspective, to querying allows the user to enter a free text query. These systems typically take the users query, Q , and apply the same indexing algorithm as was used to index the documents in the document base. The descriptor of the query is then compared against the descriptors for all documents in the document base by use of a matching function which returns a numeric value to state how good the match is. Most algorithms produce a matching value between 0 and 1, where zero states that the document does not match and one states that the document is a perfect match. Such systems typically return the results of a query as a matched document list, L , which can be specified as:

$$L = \{ \langle i, w \rangle \mid i \in \mathbb{D} \wedge w = M(D_i, \mathbb{I}Q\mathbb{I}) \wedge w \geq t \}$$

where

i = Document identifier for a (partially) matched document

w = Weight (or strength) of the matching

M = Matching function

D_i = Descriptor for document i

Q = Query

t = Threshold value below which values of M are not considered as matches

Many systems rank the results of a query so that users know that the further down the matched list they look, the less likelihood they have of finding a document that is relevant

to their query. This has two major benefits, firstly the user is presented with an ordered list so (s)he can scan the matches from the top of the list and truncate the search at any point, in the knowledge that documents which have not been viewed are less likely to be relevant than the ones already viewed. Secondly, and as a result of the first point, the retrieval engine can set the threshold, t , very low as the user is not expected to look at all matches, the choice of threshold can be left to the user to decide as (s)he scans the matched documents.

A common variation of the threshold approach is to ask users how many documents they wish retrieved, n , and guarantee that the matched document list is never longer than this. A list based on this approach could be created from L by simply taking the top n highest weighted entries. This approach has the benefit that users know the maximum number of documents which they will be presented with. However, when there are many documents with similar weights the approach will impose a cut off at an arbitrary point (from a viewpoint of document weights). For example, when a user wishes 10 documents retrieved to a query for which there are 11 documents with the same score, one of these documents must be dropped.

There is great variation in the actual algorithms which are used to perform the matching function. The simplest of algorithms counts the number of terms which co-occur in the query descriptor and the document descriptor, this can be expressed as the size of the intersection between the two descriptors:

$$M(D_i, \llbracket Q \rrbracket) = |D_i \cap \llbracket Q \rrbracket|$$

If used to produce a list of ranked matches this function would produce the same output as the expansion of a series of **anded** terms discussed earlier for boolean retrieval systems. The matching function does not take into consideration the size of the documents and will tend to give higher scores to larger documents. As an example, consider evaluating $M(D_i, D_i)$, it would be reasonable to expect that this has a constant value for all i ; however, this is not the case for the simple matching algorithm since $\forall i: M(D_i, D_i) = |D_i|$. This problem can be removed by dividing the result by the sum of the size of the descriptors. If the result is then doubled, it is guaranteed to lie between zero and one, this algorithm is known as Dice's coefficient and can be expressed as:

$$M(D_i, \llbracket Q \rrbracket) = 2 \frac{|D_i \cap \llbracket Q \rrbracket|}{|D_i| + |\llbracket Q \rrbracket|}$$

Dice's coefficient is widely used in environments which represent the document descriptor as a set (or a binary vector). It cannot, however, take any account of the weight of different terms and cannot be combined with techniques, such as inverse document frequency, which take account of words having varying abilities to differentiate between relevant and non-relevant documents. A more general, and more complex, algorithm is the

cosine coefficient, which views the two descriptors being compared as N-dimensional vectors which start at the origin of N-space, it then calculates the cosine of the angle between these two vectors. The intuition behind the cosine coefficient being that if documents *A* and *B* are very similar they should have nearly parallel vectors ($\Theta=0^\circ$, $\cos(\Theta)=1$) but if they are not at all similar then the vectors would be nearly perpendicular ($\Theta=90^\circ$, $\cos(\Theta)=0$).

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

where

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^N \mathbf{A}_i \mathbf{B}_i$$

\mathbf{A}, \mathbf{B} = the descriptors being compared

$$\|\mathbf{V}\| = \sqrt{\sum_{i=1}^n \mathbf{V}_i^2}$$

Since the cosine coefficient sums the products of each term's weight in the two documents being compared, the coefficient is very flexible and can accommodate any methods which weights terms (for example the inverse document frequency).

The last matching algorithm which will be discussed in this section was developed by Croft & Harper (1979) and is based on probability theory. Probabilistic methods of information retrieval normally require some documents to be known as relevant before they can estimate the probability of other documents being relevant to the user. This algorithm, however, requires no prior knowledge of relevance and can be used as a direct matching algorithm. When the vectors being compared have *N* dimensions the algorithm can be defined as:

$$M(\mathbf{A}, \mathbf{B}) = \frac{C P_1 + P_2}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

where

$$P_1 = \sum_{x \in N} \sum_{y \in N} \mathbf{A}_x \mathbf{B}_y$$

$$P_2 = \sum_{x \in N} \sum_{y \in N} \mathbf{A}_x \mathbf{B}_y \log \left(\frac{(N - \|\mathbf{A}\|)}{\|\mathbf{A}\|} \right)$$

C = constant between 0 and 1 defining input from each component

Experiments described by Croft & Harper show that significant improvements are made on retrieval effectiveness by using this algorithm when compared with the simple matching algorithm, an inverse document frequency base algorithm, and the cosine coefficient.

2.2.4 Relevance Feedback

Relevance feedback is a technique used in free text retrieval systems which allows the user to provide feedback to the system on the results of a query. This feedback is usually provided by the user marking documents which are relevant (and in some systems, documents which are irrelevant) as (s)he looks through the matched documents. When the user is finished looking at the current list of matched documents, a new query can be issued which is based on a combination of the previous query and the relevance feedback information. Relevance feedback allows users to refine their queries without having to reword the original query, this is a very important technique because users are often not precise enough about what they are looking for when they start a query. Even if they are sure of what they are looking for, they may have trouble expressing their requirement. Relevance feedback is also very useful because the communication between users and the retrieval engine is in terms of very high level concepts (normally entire documents); even if the system's response to feedback is low level (e.g. adjusting term weights) this provides a very natural form of interaction.

Any implementation must strongly encourage users to provide feedback, although users should learn to use it regularly if the results prove useful. Usage patterns can seriously affect the benefits of negative feedback (where users comment that documents are not relevant to the current query), users tend to ignore matched documents which are completely irrelevant and only provide negative feedback on those that almost match, but not quite. This is the opposite usage of what is required by the retrieval engine for optimum query correction (see chapter 7 for details of user testing). It can also be argued that, since the document base will probably contain many more irrelevant documents than relevant ones, giving negative feedback simply informs the system that yet another document is not relevant, whereas positive feedback states that this is one of the few documents that really is relevant. However, Ide and Salton (Salton 1971 pp. 373–393) showed that use of negative feedback could assist the next search for relevant documents, so it may be beneficial for a system to consider implementing negative feedback if the user could be educated or encouraged to use it correctly. For example, when the users are likely to become highly skilled in using the system and when there are relatively few casual users. The experiments ran by Ide and Salton were also based on a traditional query-only information retrieval system, in which the only documents the user could access were ones which the system considered as matches to the query. Section 5.4 shows that the effect of feedback varies with the strength of the original matching, and that for negative feedback, under the vector space model, the variation is non-intuitive: negative feedback on exact matches has no effect while a large effect occurs when the document did not match the query.

There are several methods for taking account of relevance feedback, the most common of which is fixed increment error correction (van Rijsbergen 1979 pp. 107). Under this method the user's query and the relevant document's descriptor are considered as N-

-dimensional vectors. When the document is marked as relevant, the two vectors will be added (with a scaling factor pre-multiplying the document vector). For term-based retrieval this results in all the terms which are used to describe that document being either added to the current query with a constant weight or, if the term already exists in the query, its weight is increased by the constant weight. This can be expressed as follows:

$$\mathbf{Q}' = \mathbf{Q} \oplus k\mathbf{D}_i$$

where

\mathbf{Q} = the query before giving relevance feedback

\mathbf{Q}' = the query after giving positive relevance feedback on descriptor \mathbf{D}_i

k = constant value between 0 and 1

$\mathbf{A} \oplus \mathbf{B}$ = vector addition of vectors \mathbf{A} and \mathbf{B}

After providing relevance feedback, possibly on many documents, it is usually necessary for users to issue a new relevance-based query. This query would use a combination of the original textual query and any feedback as the basis for matching against documents in the document base. Although it is theoretically possible for this query to be issued automatically whenever a user gives feedback on a single document, this would be highly un-desirable: the time taken to process a query is usually significant and would greatly discourage users from giving much feedback. The system would also continually re-calculate the list of matched documents preventing the user from scanning down a list giving feedback. When used in a browsing environment, as discussed in section 2.4.3, it may be preferable to make immediate use of feedback information so long as the processing time is reasonably low.

Relevance feedback is an essential part of systems which are built on probabilistic models of information retrieval. These systems rely on extrapolating known information to estimate the probability that other documents match. This relies on initially having a knowledge of some relevant (and possibly some non-relevant documents). Croft and Harper (1979) presented an initial search algorithm, which was developed with consideration of probability theory, and can be used to retrieve a list of initial matches (as indeed could any traditional matching algorithm). After the initial query the user can provide relevance feedback on some documents and this information could then be used by a full probabilistic retrieval algorithm. Van Rijsbergen (1979 pp. 111–143) presents an in depth discussion of the probabilistic model with a description of the theory behind the retrieval process. Probabilistic retrieval can, in some sense, be considered as optimal, as van Rijsbergen describes, the probability ranking principle states that “if a reference retrieval system’s response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of

the system to its user will be the best that is obtainable on the basis of those data.” In other words, using the probabilistic model to rank the matched document list will produce a list which is the best that can be achieved (for the given information). This is a very strong statement which implies that, if the matched document list is not of much use to the user, then this is because not enough information is available for the retrieval engine to base its responses on.

2.2.5 User Interfaces

Although much work has been carried out in information retrieval to improve the effectiveness of these systems to match relevant and ignore irrelevant documents, very little work has taken place to improve the user interface to such systems. This is unfortunate because many of the problems in information retrieval can be reduced by a well designed user interface. For example, problems concerning the length of the matched document list can be reduced if the user can easily scan through this list and decide which documents to pursue. The use of natural language queries and relevance feedback are the only two widely used methods which were developed through consideration of user interaction. The remainder of this sub-section provides a short discussion of systems which have given considerable consideration to the user interface.

Thomas

Thomas is a retrieval system developed by Oddy (1977) which, although being entirely text driven, provides a very advanced user interface. Users start their search with an initial query, the system then presents the best match to this query together with the authors and index terms associated with the query. Users can then provide relevance feedback (by saying ‘Yes’ or ‘No’), and they can also state that specific terms or authors are of interest (or indeed that certain terms/authors should be rejected from matches). A sample query session is listed below with user commands shown in bold typeface.

Start Searching:

> **pulmonary alveoli**

Influence of fasting on blood gas tension, pH, and related values in dogs.;
Pickerell *et al.*, *Am J Vet Res*, 34, 805-8, Jun 73

1. J A Pickerell, 2. J L Mauderly, 3. B A Muggenburg, 4. U C Luft, 5. animal experiments, 6. animal feed, 7. arteries, 8. blood, 9. body temperature, 10. carbon dioxide, 11. dogs, 12. fasting, 13. hemoglobin, 14. hydrogen-ion concentration, 15. irrigation, 16. lung, 17. oxygen, 18. pulmonary alveoli, 19. respiration, 20. time factors

> **No, 10,17,19,20**

The arterial-alveolar nitrous oxide...

> **Yes**

The user initiates a search by entering a very simple query which the system matched to the paper by Pickerell. The user then stated that this is not relevant (**No**) but that certain aspects are (i.e. terms carbon dioxide, oxygen, respiration, and time factors). The user could also have stated that the document is relevant (by saying Yes) or that certain terms are not relevant by prefixing them with the word not (e.g. **12,18, not 11,13** states that 12 and 18 are useful but we are not interested in 11 or 13), or introduce new terms to widen the query. Although rather dated and a text-only interface the extensive use of query reformulation through the human-computer dialogue separates this system from other text driven retrieval engines. The system also provides on-line help to give brief descriptions of what commands may be used (often including examples) which may be called up whenever the user is faced with a prompt.

Caliban

Caliban (Frei and Jauslin, 1982) provided a major step forward in user interface design for information retrieval systems. It provided the first significant window and pointer based user interface. As well as providing form based querying Caliban allowed users to browse through the structure of the document base (e.g. through the thesaurus). When creating queries users are also able to browse through the document base structures to find useful items to search for (e.g. terms from the thesaurus).

I3R

Thompson and Croft (1989) describe a retrieval system which presents the user with a pointer (e.g. mouse) based interface which makes extensive use of browsing techniques. A description of the system is given later in section 2.4.

NRT

The News Retrieval Tool, Sanderson and van Rijsbergen (1992), which uses newspaper archives as a document base and provides an almost entirely mouse-based interface to free text retrieval. The only time users are forced to use the keyboard is to enter the initial string which is then analysed by the system, a list of terms is displayed in a window together with weights which are presented as sliders and are calculated using inverse document frequency (the user can later alter these weightings by direct manipulation). The search then displays a retrieval window which contains a list of matched document records (article icon, article title, source newspaper, and article size) in decreasing order of matching score, the user can now open any of the matched records to display the content of the newspaper article. Relevance feedback is also implemented as a mouse driven activity, to state that an article is relevant to the current query feedback the user drags the articles record from the retrieval window into a relevant articles window which records that

article as being relevant. At this stage the article's icon changes to remind the user that the article has been marked as relevant. The retrieval window overview of the matched document list would provide an elegant solution to access problems associated with the hybrid user interface discussed later in this thesis (see chapters 6 and 7).

Apart from user interface design issues, NRT provides a powerful separation between the interface and the underlying retrieval engine: the interface runs on a personal computer (an Apple Macintosh) while the retrieval engine can be run on various platforms. The remote platforms include accessing a newspaper archive service by modem and a workstation connected by a local area network. As the newspaper archive service did not provide relevance feedback, this was implemented as part of the user interface.

A very similar interface and methodology, of user interface to information server connection, was presented by Stein (1991) for the WAIS project. The user interface presented in the paper models a question to the retrieval engine as: a query string, a set of sources which will be accessed, a set of relevant items, and, after the query has been executed, a set of matching documents. Users can provide feedback by dragging documents from the results section into the *similar to* section of the same question window, or by dragging the matched document to a new question window, which starts a new question based on the relevance information only. Paragraphs of text can also be marked as relevant by dragging the selected text into the *similar to* section of a question window. Although the interface is significantly better than many other information retrieval interfaces, the most significant aspect of the WAIS project is the provision of a standard server protocol (an extension of the NISO Z39.50 standard) so that different document bases in different sites can be accessed using a WAIS compatible front end. There are already many sites, mainly in the U.S.A., which operate experimental WAIS servers.

Rabbit

One system which provides access to databases (as opposed to unstructured document bases), which has a relevance feedback based interface, is Rabbit (Williams 1984). Rabbit provides a system in which databases can be queried within a mouse-based environment and makes extensive use of relevance feedback, or as the authors term it *critiquing*. A brief initial query is re-formulated by use of six critique commands: Require, Prohibit, Alternatives, Describe, Specialize, and Predicate. These commands allow the user's query to become more tightly specified until a suitably sized list of matches is presented. Rabbit shows that the process of query by reformulation can be automated in traditional databases, as well as for access to unstructured documents.

2.2.6 Evaluating IR Systems

Information retrieval systems have traditionally been evaluated by the use of two measures: *precision* and *recall*. To formally describe these measures we require three

values: the number of documents which the user considers relevant which are in the list of matched documents, m , the total number of relevant documents in the document base⁶, M , and the size of the matched document list, L . Recall is defined to be the fraction of relevant documents which are actually matched by the retrieval system, m/M , and precision is defined as the fraction of documents which the system matched which are actually relevant, m/L . A perfect retrieval system would have precision and recall of 1, and would thus match all documents which the user would consider relevant and no others. This is, however, not achievable by automatic retrieval systems – one reason being that the user never gives the system all the information which (s)he subsequently uses to decide on relevance. Information retrieval engines typically have a average precision-recall graph as shown in figure 2.1. As recall is increased so that more relevant documents are matched, the number of irrelevant documents which are matched also increases, thus reducing the precision.

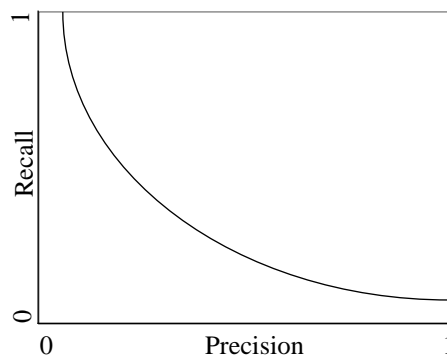


Figure 2.1: Average Precision-Recall Curve

The curve shown on figure 2.1 shows that perfect precision and perfect recall are possible, but in reality this is not the case. Perfect recall could be achieved trivially by matching every document in the document base but this is equivalent to the user simply looking through the document base manually. Perfect precision, on the other hand, cannot be guaranteed by any automated system since it is impossible to guarantee that a retrieval engine can find any of the documents which are actually relevant. An alternative to precision and recall is given by the combined E measure (van Rijsbergen 1979 pp. 174–175) which gives a single measure of retrieval effectiveness.

When used in a retrieval engine which ranks the list of match documents, the terms precision and recall have to be re-defined to take account of the ranking. When a system orders the output from the retrieval engine by rank, it can provide the user with many more matches and transfer the imposition of a cut-off to the user – who will simply stop searching down the ranked list when the matches become poor. Since the standard definitions of recall and precision are based on the number of documents retrieved, an adjustment must be made to consider how close to the top of the ranked list the documents

⁶ whether or not they have been retrieved

are. This takes some account of the likelihood that the documents will appear above the user imposed cut off point, and are thus seen by the user. If such a variation of recall and precision were not made, the values of precision and recall for ranking systems, which provide many matches, would not satisfactorily reflect their effectiveness. The high number of matched document would increase the recall of the system, but drastically reduce the precision.

The measurements discussed so far in this sub-section are the most commonly used methods for judging the effectiveness of an information retrieval system. However, they only cover one aspect of the retrieval system - its ability to distinguish relevant from non-relevant documents. While this is important, many other factors affect the usability of a system, which in turn alters the effectiveness of retrieving useful information. Cleverdon (1966) presented a list of six criteria for evaluating information retrieval systems, this list is composed of precision, recall, and the following criteria.

Coverage

The fraction of documents on a potential subject which are stored within the document base. For most domains it is impossible to accurately state this value, but it can usually be approximated by considering the fraction of periodicals in the subject which are on-line, whether theses are also stored, and various other factors. An impression of the coverage of a document base can be given by simply stating the number of documents in the document base within a particular domain. Users will only find a retrieval system helpful if it can produce information on a significant amount of the documentation which is produced in that field.

Time lag

The time lag or response time of the system can seriously affect the user's ability to retrieve useful documents. For example, users will be very reluctant to re-phrase a query or use relevance feedback if the operation takes the order of minutes to execute. Definite rules on what time lag is acceptable are hard to define but it is clear to users of a system when the lag time is too long. When using a search intermediary (e.g. a librarian trained for on line searching) the time lag is considerable, possibly a matter of days elapse between the query being given to the intermediary and the results being received. In this circumstance the search intermediary is trained in interview techniques so that the query, once agreed, does not require modification and the user should not have to submit a second query. If the system is less effective at extracting the user's information requirements, and at finding suitable matches, the time lag will have to be considerably shorter to permit reformulation.

Presentation issues and user effort

There are two closely related criteria for evaluating a retrieval system. Presentation issues are very widespread and range from major design decisions of the system, such as the

paradigm of choice (e.g. direct manipulation), through to very small aspects of the interface's graphical design. It is through the presentation of a system that many of the benefits of good user interface design can be brought to bear to produce an easily usable system, which should lead to more effective use of the system. The effort required by a user is also strongly related to user interface issues, in that the harder the user must work to get at the required knowledge the less likely (s)he is of using the retrieval system effectively. If the interface is poor enough users will not be able to use the system at all, or will not persevere long enough to learn how to use the system. Tague and Schultz (1988) describe many aspects of the evaluation of user interfaces to retrieval systems, their list includes measures on the friendliness of the system and how informative retrievals are. They also describe procedures which can be used to evaluate the user interface. A further discussion of user interface issues concerning information retrieval systems is given in Chapter 7.

2.2.7 Multimedia Access

Since document descriptors are accessed during a search, rather than the document content, information retrieval does not have any direct requirement for the documents to be composed of text. A serious problem, however, occurs when considering the calculation of these descriptors. In a textual environment, words and phrases are suitable sized elements which permit retrieval based on descriptors, for non-textual documents it is very difficult to envisage a suitable set of objects to use in the creation of descriptors. One might consider that pieces of text describing the objects in an image may be used to retrieve pictures, however, this would rely either on full automatic image interpretation which is not, currently, possible or on human indexing which is not only slow, but as Enser (1991) showed, error prone. The problem becomes more acute when considering other computer presentable media, for example music, in which no suitable indexing terms can be considered which are not based almost entirely on emotion (impression) or technical content. There are, however, certain classes of all media which are amenable to direct indexing. These are non-textual media which naturally carry a textual equivalent: e.g. the script of a movie or television programme, the words of a song, the words of spoken speech, or the text within a picture. While entirely computer-based indexing of these sub-media may not yet be possible, the human input is very well defined and not extensively error prone. Some novel solutions to the retrieval of non-textual objects are discussed in section 2.5.

2.2.8 Large Scale Usage

Free text retrieval is designed to be used on a large scale, and has been refined over many years to produce good quality and reasonably fast search algorithms (for example, without much consideration of speed during development, *mmIR* (described in chapter 6) searches

approximately 125 paragraphs per minute on an Apple Macintosh Classic⁷, and approximately 940 per minute on a more powerful Macintosh IIfx⁸). With increasing document base size many techniques can be used to improve speed. The simplest improvement being the use of inverted files. For each term used in the document base an inverted file keeps a list of documents in which that term exists, thus a query need only search those documents which contains one of the terms used in the query.

There has also been work into exploiting the benefits of parallel computers for index file matching (e.g. Waltz, 1987). The matching process of free text retrieval is ideal for exploiting any available parallel processing, there are many reasonably small operations with very little communication. The matching algorithm has to be executed on a large number of descriptors, this process takes a short (but non-trivial) period of time, and each calculation returns a single real number.

When inverted files fail to provide a fast enough search engine, the search can be speeded up further by initially splitting the document base into two categories: potential matches and definite non-matches. This split can be made using a fast algorithm which guarantees to correctly allocate all documents which would pass the full matching algorithm, but the algorithm may wrongly assign some documents which will fail the full matching algorithm.

Clustering of the document base can provide a highly significant increase in matching speed. Clustering techniques are used to group sets of documents together which have very similar content, each cluster then has a cluster representative calculated which *averages* the meaning of the documents in the cluster. In a clustered document base queries are matched with the cluster representatives rather than with individual documents, resulting in a much faster search time since significantly fewer comparisons are required with the cluster representatives than would be required if every document were searched. The members of the clusters which match the query can then be individually matched if required, so that the user can be presented with a list of matched documents (as opposed to matched clusters). Croft (1978) presents a review of clustering algorithms together with his own work on access to large files of documents by clustering techniques. The problem of calculating clusters can be split into two sub-problems: classification and descriptor calculation. Classification addresses the problem of deciding which clusters should exist, and which documents should be included in which clusters. Van Rijsbergen (1979 Chp. 3) discusses automatic classification at length, he states that the process of clustering is based on the hypothesis that “closely associated documents tend to be relevant to the same requests”.

Once the clusters have been created and documents have been classified to each cluster, a descriptor which encompasses the information held in the documents of the cluster must be calculated. Section 3.3 describes a common cluster descriptor calculation algorithm –

⁷ An 8 Mhz 68000 based personal computer.

⁸ A 40MHz 68030 based machine.

the cluster centroid. This algorithm relies on documents being modelled as vectors of real values and calculates the point in space which is at the centre of the points represented by the document vectors. It can be described as follows:

$$R = \sum_{\forall \mathbf{D} \in C} \frac{\mathbf{D}}{\|\mathbf{D}\|}$$

where

C = set of document descriptors in the cluster

$$\|\mathbf{D}\| = \sqrt{\mathbf{D}_1^2 + \mathbf{D}_2^2 + \dots + \mathbf{D}_N^2}$$

N = number of different index terms (i.e. the dimensionality of \mathbf{D})

When clustering techniques are used in a boolean vector retrieval system a variation of the cluster centroid algorithm can be used. Instead of calculating the mean weight for each term in the cluster the algorithm can be altered to assign the term to the cluster representative if it occurs in a suitable number of documents. Van Rijsbergen (1979 pp. 99–103) describes the calculation of cluster representatives in a boolean environment in greater detail.

2.3 Hypermedia

Hypermedia⁹ approaches searching for information from a completely different angle than query based free text retrieval. Users are typically given initial access to index nodes (or header nodes) which contain general areas which are covered by the document base, the user selects one of these areas and is presented with either a document or another index node. The distinction between index nodes and documents is rather weak as any document can contain connections to any other documents or index nodes, these connections are known as *links*, and the general term *node* is used to refer to both index nodes and documents. Many hypermedia systems do not contain traditional documents but are composed of many small nodes which are linked together to form larger sections, which the user can browse through by following links. A set of academic papers on a related subject could be considered as forming a paper-based hypermedia network since citations provide links between the documents (or nodes), and papers often have internal links, e.g. “see section 1B for further details”. Although a set of academic papers can be considered a mild form of hypermedia, the key feature of hypermedia systems, which distinguish them from citation following systems, is the separation of structure from content. Users can use the structure which is given by the document’s author, or they can use the units of information in the document as components which are connected with their own structure.

A hypermedia-style system was first envisaged by Bush (1945), and initial work on a computer-based implementation was started by Nelson (1967) and Englebart (1963).

⁹ Throughout this thesis the term *hypermedia* will be used as a general term encompassing the text only (or *hypertext*) and multimedia variants of the access method.

Despite this long history, the area received very little attention, with the exception of a few research projects, until the late eighties and early nineties when it was seen as the natural access method for document bases of mixed media which were starting to be developed.

Bush described a fictional system, called the Memex, which would be able to connect pieces of information together to form paths through the document base. These paths would be similar in nature to trails of association which human memory can make while trying to remember faded information, and they need not follow any specific guide-lines concerning the relationship between the nodes. Bush envisaged the Memex being constructed by the use of automatic, high-speed microfiche readers which would store links to other nodes as encodings which could be read by photocells. Despite the paper being written in the very early days of computer technology, the Memex encapsulated many of the concepts which form the basis of modern hypermedia systems: fixed size fragments of information which are linked together by static, user defined links. He envisaged users being able to obtain microfiches from external sources (e.g. dealers or colleagues) with some paths already constructed, but with the facility for users to add their own paths through the data.

The first computer based hypertext system was considered by Nelson (1967), who also coined the terms *hypertext* and *hypermedia*. He defines hypertext as “the combination of natural-language with the computer’s capacities for interactive, branching or dynamic display, *when explicitly used as a medium*”, or more broadly “as a generic term for any text which cannot be printed (or printed conveniently)” and he considers that “‘non-linear text’ might be a fair approximation”. The definition of hypertext as a medium is important, Nelson points out that it is this element of the definition which differentiates hypertext (and hypermedia) from systems which simply provide rapid look-up and cross-referencing. Nelson also compares the state of hypertext in 1967 – the discussion is equally valid today – to the early days of the motion picture industry in which many techniques of the closest previous medium, stage plays, were imported unaltered with very poor results.

Nelson is also the driving force behind the Xanadu project (Nelson 1990) – a project which plans to develop a global receptacle for documentation. It was to this end that much work has been carried out on compacting documentation through the use of links: if a document includes a portion of another document then it should be included by link rather than textually. The project has also carried out much work into the setting up of automatic royalty and copyright controls so that Xanadu can be used in a commercial and legally protected world.

Conklin (1987) presents a general introduction to hypermedia in which he categorises the essential components which are required of a system to justify being referred to as a hypertext / hypermedia system. The paper also presents a survey of 18 hypermedia systems which were available at the time of publication, these include all the research systems on which the ideas of hypermedia have been developed, and discusses the

strengths and weaknesses of hypermedia. Begoray (1990) presents a more recent survey together with an overview of the design decisions involved in creating a hypermedia system and definitions of many of the terms used in hypermedia. Begoray concludes with a call for research into a model for the complex cognitive tasks in which readers and authors of a hypermedia system are involved. Adams (1990) provides a more casual introduction to hypermedia, and Neilsen (1990) presents a wide ranging review of the hypermedia techniques, implementations, and applications of the medium.

This section initially presents a simple model of hypermedia, it then goes on to discuss extensions and alternatives to this model, and finally some consideration of the usability issues affecting hypermedia systems.

2.3.1 General Description

Hypermedia document bases can be considered as a set of interconnected nodes, each node being composed of a content and a set of nodes which can be reached from it. The content of a node is the text (or other media) which the user is presented with when the node is displayed, this may be composed of a very small piece of information or a long document. Using a variation of standard set notation the document base can be defined as below, the most significant deviation from standard notation, is the use of $x \rightarrow y$, to state that a path (series of links) exists between x and y , and of $x \rightarrow^1 y$ to state that a direct link exists between x and y .

Base	= { $N_i \mid i \in \mathbb{D}$ }	<i>all accessible nodes</i>
where		
\mathbb{D}	$\subset \mathbb{Z}^+$	<i>currently valid nodes identifiers (integers)</i>
N_i	= $\langle C_i, L_i \rangle$	<i>pair of content and links from this node</i>
L_i	= { $j \mid j \in \mathbb{D} \wedge N_i \rightarrow^1 N_j$ }	<i>indexes which are linked to, from N_i</i>
C_i		<i>the content of node i</i>

This definition provides a very simple model of hypermedia in that it only provides one type of link and one type of node. The model is based on directed graph models similar to those proposed by Garg (1988) for use in defining concepts such as aggregations and generalisations, and Tompa (1989), as a prerequisite of his own work on providing some of the functionality of record databases to hypermedia databases.

To guarantee access to all nodes it must be shown that, assuming H is the home node from which the users' browsing commences, $\forall i \in (\mathbb{D} - \{H\}) : N_H \rightarrow N_i$. This states that a path exists between the home node and all other nodes in the document base. In some hypermedia systems the user does not always start browsing from the same point but chooses a starting point (e.g. from the underlying file system). In these systems the single home node would be replaced by a set of home nodes and the access condition rewritten

as follows: $\forall i \in (\mathcal{D} - H): \exists h \in H. N_h \rightarrow N_i$ where H is the set of possible starting points for a users browse.

2.3.2 Link Types

The links between nodes in the above example are limited to only one type, many hypermedia systems provide various types of links to connect nodes. Alternative link types can be used to control the display characteristics of the destination node (e.g. whether it should be displayed in a separate window or expanded within the current text). Different types of link can also be used to annotate the hypermedia network, for example links created by users could be differentiated from links which are created by the system and links created by the current user may be emphasised.

One use of typed links would be to form paths through the document base. These links would provide a pre-defined path through the document base which users could follow, or choose their own browsing pattern. Paths are useful in an environment where the information can be naturally structured in a linear fashion or where many nodes require prior knowledge, which is given in previous nodes along the path. Paths could also be used to define a main trail of topics through the document base, while users are left to browse each topic freely.

Another form of typing links could be to weight the links so that users have an idea of how strongly the creator of the link considered that it was worth following. This could be used as a method of producing paths, in that the recommended route is strongly weighted. Consideration would have to be given to regulating the weighting of links to prevent users from entering spurious strongly weighted links. Some work in free text retrieval has considered the use of relevance feedback to enhance the descriptors of nodes in the hope that future searches are of better quality. An analogy in hypertext would be to increase the weight of a link every time a user follows that link. Pausch and Detmer (1990) show that the provision of node popularity information does affect the choice users make for which node to follow, and it does reduce the tendency to choose the first or last node in preference to other nodes. Without the knowledge of whether the user actually found the link useful this technique may, however, not provide an improvement in retrieval effectiveness. It may also result in the first path that anyone followed becoming predominant since future users will tend to choose it first (further increasing its weight) before trying other paths.

The definition of hypermedia given in section 2.3.2 more naturally defines static links, links which do not change with time but may only be created or destroyed. Some systems also provide dynamic links which can change their destination as some function of the environment, an example use could be a link to today's newspaper. One very powerful form of dynamic links is used in the Macintosh based HyperCard system developed by Apple (1987), in this system all objects which can be selected may have a script (or small program) associated with them. This script can be executed when the object is clicked and

can then decide to create and follow a link to another node, example uses are “go to card ID 1234” which is the HyperCard equivalent of a static link, and “go to card “Newspaper” && the Date” which takes the user to today’s newspaper. Scripts may also be used to perform other operation, such as animation and calculations.

So far links have only been considered as connecting entire nodes together. Most hypermedia systems, however, provide point-to-node links. For example, Guide (Brown 1986) provides links from any piece of text. It is also possible for some systems to provide point-to-point links. These links allow the end(s) of a link to be specified as a certain *area* of the source (or destination) of the link, an area could be defined as any piece of the node which can be selected by the user, for example a word, phrase, or paragraph in textual prose, a rectangle in a raster image, or a scene in a movie. While it may be possible to emulate this kind of linking by use of smaller nodes it is more natural to provide links which are of finer granularity than the node-to-node links which the model in this section is based on.

2.3.3 Node Types

Like links, nodes can also be typed, however, typed nodes are not as widely used. The only examples in a widespread hypermedia system are the file boxes of NoteCards (Halasz 1988). These are specialised cards which are used to help users structure their hypermedia document base, they are composed of a set of links to other cards (file boxes or note cards) but contain no data of their own. NoteCards also presents maps of the system as specialised cards, browsers, which the users manipulate in the same manner as other cards.

2.3.4 Alternative Models of Hypermedia

One alternative to the directed graph based model of hypermedia has already been discussed, namely the entirely calculated, object-oriented model which is used in HyperCard. This model considers each node in the document base as a card in a stack. Implicit static links are created between each node and its predecessor, successor, the first card in the stack, and the last card. All other links are calculated dynamically as the result of a command in the HyperTalk language which specifies the destination of a node by card number/name and/or by stack number/name (there is also a visual linking mechanism which produces HyperTalk commands, so that users need not be exposed to the programming language). Since only the implicit links within a stack are static, it is not sensible to model the dynamic structure of HyperCard as a directed graph.

An alternative to the directed graph model was proposed by Stotts and Furuta (1989) and used petri-nets to define the structure of the document base and give limited control over browsing in the hypermedia network. A petri-net consists of a set of *places* (which are similar to nodes within a directed graph), a set of *transitions* (similar to buttons within a directed graph hypermedia system), and a set of *arcs* which connect transition and

places together (links). Petri-net based systems provide not only the structure of the document base but also some control over the semantics (or browsing behaviour of the system's users). Each place in the petri-net has a *marking* associated with it (the marking is a non-negative integer, often restricted to zero or one), in a hypermedia system this could be interpreted as stating that the user is currently viewing this node. A transition can only be *fired*, or activated, if all the places which point to that transition have a non-zero marking. When a transition is fired the marking of each predecessor is decremented by 1, and the marking of each successor is increased by one. A transition (and associated arcs) could be considered as a button which takes the user from one set of nodes to another set.

The diagram in figure 2.2 shows an example petri-net and the flow of markings through it. When a user enters this section of the hypermedia network (s)he is initially presented with the document which is held at place $s1$, together with two buttons (or transitions) $t1$ and $t2$ (see figure 2.2 part A). In this case the user has chosen to activate the button $t2$ which results in the removal of place $s1$ from the display and the display of both $s3$ and $s4$ (figure 2.2B). At this stage only one button is available, $t3$, and the user chooses this – note that button $t4$ is not available because place $s2$ has a marking of zero, therefore the transition $t4$ cannot be fired. When the user selects button $t3$ the place $s4$ is removed from the display and replaced by node $s5$, with $s3$ still displayed (figure 2.2C). Finally only the button $t5$ is available which takes the user to $s7$ (figure 2.2D). When the user initially chose button $t2$ in preference to button $t1$, (s)he was restricted to eventually reaching $s7$ even though, when considering the network as a graph based tree, both $s6$ and $s7$ are children of $t2$.

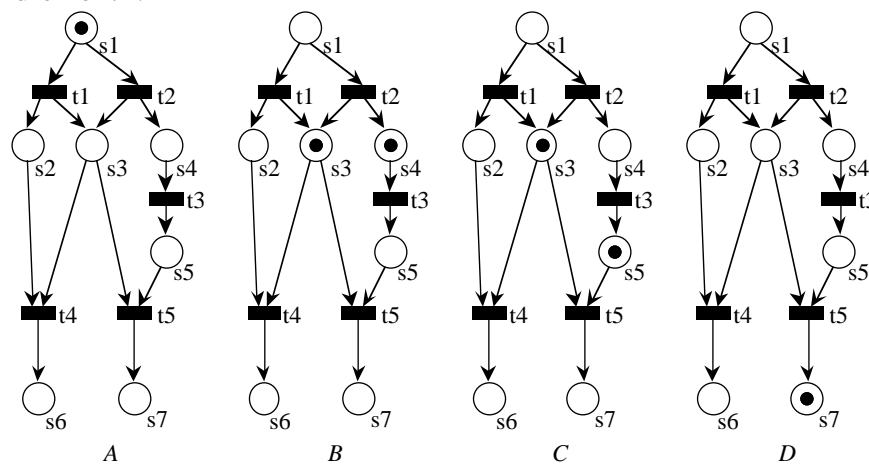


Figure 2.2: A Sample Petri-Net

Although petri-nets provide a level of control over the user's browsing they are difficult to formulate. Furuta and Stotts (1989) provide many useful examples, however it is not clear whether the technique of using petri-nets for hypermedia can be scaled up to large or very large document bases.

2.3.5 Conversion From Traditional Texts

Traditional documents can easily be converted into a weak form of hypermedia if there exist cross references in the original document(s), the task is much harder when attempting to create a well connected, true hypermedia network. Raymond and Tompa (1988) describe many of the problems which may occur and decisions which must be made concerning the conversion of text to hypertext, with detailed reference to their implementation of the Oxford English Dictionary. They state that “from the point of view of document conversion, hypertext’s main character is *fragmentation*”, and that the fragmentation of a text document into “discreet and independent” pieces of content or structure is the major problem facing the conversion of text to hypertext. Firstly, suitable size fragments must be chosen to provide pieces of the document which may be read independently, this is complicated by variation in size of sections within traditional text, for example in the Oxford English Dictionary 5% of the entries amount for 48% of the volume of the dictionary. The fragmentation of the original document(s) is performed to make the implicit structure of the paper document explicit in the hypertext version, Raymond and Tompa postulate that “the key question in conversion must be: will an explicit structure be as expressive as the implicit structure [for the given document]?”; if this is the case then the non-hypertext document would benefit from being converted into hypertext form.

2.3.6 Navigational Issues

In large document bases much care must be taken in the provision of techniques to allow users to understand where they are in the network. Various solutions have been proposed to the problem of “getting lost” in the document base, and this sub-section will briefly overview the various solutions.

History – the simplest form of navigation aid is the provision of a history command so that users can backtrack along the path that they have followed until they reach a point that they recognise. Akscyn, McCracken, and Yoder (1988) claim that the provision of very fast node access combined with a very fast backtrack command, solves many of the navigation problems typically experienced by users.

Maps – diagrammatic representation of the hypermedia network. These can be useful for users to work out where they are with respect to fixed points in the network which they know, or with respect to the path that they have already followed. Systems using maps to give assistance to the user include gIBIS (Begeman and Conklin 1988) and NoteCards (Halasz 1988). Maps are, however, not easily implemented since they attempt to map the naturally multi-dimensional hypermedia network onto two dimensions. The hypermedia network is composed of various nodes interconnected by various link; this potentially chaotic linking strategy which creates a more complex network than a traditional hierarchy also prevents any convenient 2-D version of the network being drawn. Many of the issues

involved in drawing maps which are of use to end users are discussed by Hofmann *et al.* (1991).

2.3.7 Evaluating HM Systems

Very little work has been done on evaluating the usage of hypermedia systems with respect to the quality of search results. The seven evaluation factors which are used for information retrieval, with the exception of precision and recall, can be used directly in a hypermedia environment. Because of their dependence on a computer generated list of matched documents, the terms precision and recall do not naturally translate to a browsing environment (in which no such list exists). The remainder of this section discusses an alternative pair of criteria for which a hypermedia document base can be tested against – these measure are, however, not directly comparable with precision and recall. Therefore hypermedia document bases cannot be directly compared with free text bases using these criteria.

A measure *resistance* is defined here to give a feel for the quality of the access paths to nodes which are relevant to the current information search. This measure is defined as approximating the resistance of an equivalent electrical circuit; each node being given a resistance which depends on the ratio of useful to useless links exiting from it, and each link being modelled as a wire with zero resistance. The resistance of node x along a path to node y can be defined as $(L_x - N_{x,y})/L_x$, where L_x is the number of links exiting node x and $N_{x,y}$ is the number of links exiting node x which eventually reach node y . This definition of a node's resistance takes into account the number of choices from which a user can pick a path which will lead to the destination node. For example, figure 2.4 shows the equivalent circuit for the hypermedia network shown in figure 2.3.

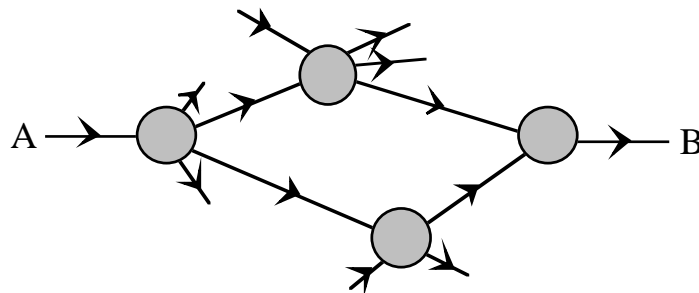


Figure 2.3: Example hypermedia network.

Here the user is assumed to be travelling between points A and B . Nodes are shown as grey circles connected by directed lines, arcs. Arcs which are not connected to a node in figure 2.3 are considered to be connected with nodes which are not on a path from A to B . This network is equivalent to the following circuit, in which resistors are shown as boxes connected by straight lines, zero resistance wires.

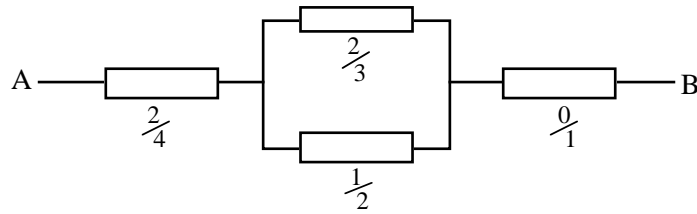


figure 2.4: Electrical equivalent of example hypermedia network

There are two methods for calculating the resistance of simple circuits. All sequential components can simply be added together. Components which are in parallel are combined with inverse addition: each resistance is inverted, the inverted resistances are added, the total is then inverted to produce the resistance of the parallel section. Using these rules the example circuits has an overall resistance of:

$$\frac{2}{4} + \frac{1}{\frac{2}{3} + \frac{1}{2}} + \frac{0}{1} = \frac{1}{2} + \frac{2}{7} + 0 = \frac{11}{14} \approx 0.79\Omega.$$

The definition of resistance for a path, $R'(x,y)$, takes into account the number of alternative paths from x to y and the difficulty of following each of these paths, while the resistance of a set of destination nodes (e.g. the set of relevant nodes) is defined as the average resistance to each member in that set and defined as $R(x,Y)$. The notation used below includes $x \rightarrow y$ which states that y can be reached from x , $x \rightarrow^1 y$ which states that y can be reached from x by following precisely one link, and $x \nrightarrow y$ which states that y cannot be reached from x .

The following table of expressions defines the operations used above for simple circuits in terms of recursive functions. It contains three columns: the first column contains the expression being defined (e.g. $R(x,Y)$), the second column specifies the definition of that expression under the condition that the expression in the third column holds.

$$\begin{aligned}
R(x,Y) &= \frac{\sum_{y \in Y} R'(x,y)}{|Y|} & x \in \mathbb{D} \wedge Y \subset \mathbb{D} \\
R'(x,y) &= 0 & x \in \mathbb{D} \wedge y \in \mathbb{D} \wedge x=y \\
&= \infty & x \in \mathbb{D} \wedge y \in \mathbb{D} \wedge x \not\rightarrow y \\
&= \frac{L_x - N_{x,y}}{L_x} & x \in \mathbb{D} \wedge y \in \mathbb{D} \wedge x \rightarrow l_y \\
&= \frac{L_x - N_{x,i}}{L_x} + R'(i,y) & x \in \mathbb{D} \wedge y \in \mathbb{D} \wedge x \rightarrow l_i \wedge i \rightarrow y \wedge N_{x,y}=1 \\
&= R'(x,i) + \frac{L_i - N_{i,y}}{L_i} & x \in \mathbb{D} \wedge y \in \mathbb{D} \wedge x \rightarrow i \wedge i \rightarrow l_y \\
&\quad \wedge N_{i,y}=1 \\
&= \frac{L_x - N_{x,y}}{L_x} + \frac{1}{\sum_{i \in I} \frac{1}{R'(i,y)}} & x \in \mathbb{D} \wedge y \in \mathbb{D} \wedge N_{x,y} \leq P_{x,y} \\
&\quad \wedge I = \{i \mid x \rightarrow l_i \wedge i \rightarrow y\} \\
&= \frac{1}{\sum_{i \in I} \frac{1}{R'(x,i) + R'(i,y)}} & x \in \mathbb{D} \wedge y \in \mathbb{D} \wedge N_{x,y} > P_{x,y} \\
&\quad \wedge I = \{i \mid x \rightarrow i \wedge i \rightarrow l_y\}
\end{aligned}$$

where

$$\begin{aligned}
L_i &= |\{n \mid i \rightarrow l_n\}| & \text{number of links from } i \\
N_{i,j} &= |\{n \mid i \rightarrow l_n \wedge n \rightarrow j\}| & \text{number of successors of } i \text{ which eventually} \\
&\quad \text{reach node } j \\
P_{i,j} &= |\{n \mid i \rightarrow n \wedge n \rightarrow l_j\}| & \text{number of predecessors of } j \text{ along paths} \\
&\quad \text{from } i
\end{aligned}$$

This definition is equivalent to the resistance of an electrical circuit so long as the circuit is reasonably simple (i.e. the circuit can be broken down into entirely serial or parallel sub-circuits). For more general circuits, which cannot be solved using the simple techniques for serial or parallel circuits but must be solved using Kirchoff's laws, this algorithm provides an approximation to the resistance. This approximation is still a useful measure of how difficult a user will find browsing between the two end nodes. The model does, however, consider all links as having a resistance of zero. This can lead to some links between nodes being ignored (e.g. when the two nodes are linked by a direct link and some other route, the zero resistance *short circuits* all other paths between the two nodes). This problem could be solved by considering all links to have a constant non-zero

resistance, but this would significantly increase the complexity of the resistance algorithm with little improvement in general accuracy.

A measure of *tightness* or connection can be defined to give a feel for how likely a user is to find all relevant nodes once one node has been found. This value could be defined to be the average resistance between any two members of the cluster:

$$T = \frac{\sum_{\langle x,y \rangle \in P} R(x,y)}{|P|}$$

where

$$P = \{ \langle x,y \rangle \mid x,y \in C \wedge x \neq y \}$$

C = the cluster of nodes being considered, e.g. the set of relevant nodes

It should be noted that these measures are at their best when the value is smallest and that there is no maximum value of resistance or tightness. These factors also do not share the natural inverse relationship that precision and recall do in a query based retrieval environment.

Aigrain and Longueville (1991) present a calculation for the *probabilistic distance* between two nodes in a hypermedia network. This measure is based on the question: “what probability has he [the user] to find the target image in a given number of steps?”, and is defined as the minimum number of steps which yield a probability no less than a specified value, p . The probabilistic distance can be defined in terms of p as:

$$dp(x,y,p) = \min (\{ m \mid \text{prob}(x,y,m) \geq p \})$$

where $\text{prob}(x,y,p)$ is defined as the minimum number of steps for which the probability of the user leaving x and reaching y in that number of steps is greater than p . Interesting values for p are 0.5, the minimum number of steps which gives a fifty percent chance of the user finding the node within the number of steps, and $p=0.9$ which gives a reasonable approximation the maximum number of nodes which the user will need to access to find the given node. To guarantee access within a specified number of nodes (i.e. $p=1$), would result in the size of the document base being returned as the minimum number of steps to guarantee this criteria, since scanning every node is the only approach which guarantees that the user will find the required node. The probabilistic distance can be most accurately measured with real users but can be approximated to using a given model of how users select which links to follow, the model of selection can then be evaluated by comparing results with real user tests. The description given above for resistance is similar to the probabilistic distance when it is assumed that all nodes have equal likelihood of being selected. The probabilistic distance can be made more accurate by considering selection strategies of users, whereas the resistance measure is fixed for a given document base and

can be calculated from the structure of the hypermedia without requirements to consider such strategies.

2.3.8 Multimedia Access

Since the hypermedia model of information retrieval does not access the nodes except for display purposes any media can be accessed by the model. In the basic model described in section 2.3.1 the only new code which is required when a new medium is added is a method to display nodes in the new medium. Most systems, however, provide the source of links as areas within nodes, therefore when adding a new medium, a method of creating the source of links (and possibly of handling destinations as sub-node elements) would also be required. Hypermedia systems can easily be designed so that an extensible set of media can be accessed and are often used to access text, images, sound, and video.

2.3.9 Large Scale Usage

Hypermedia systems are often used for small document bases (for example many bases have fewer than a hundred nodes). When the document base becomes large, the requirement for users simply to browse to the information they require becomes impractical. The authoring process also becomes very difficult as the document base increases in size: when authors enter new nodes in a hypermedia network they must, theoretically, scan every node in the network to assess where links should be provided. The hand crafted links and browsing in hypermedia networks are designed for small to medium sized document bases and simply cannot be expected to work in networks containing millions of nodes.

2.4 Combining Free Text Retrieval and Browsing

The previous two sections have described the two major areas of research into provision of access to document bases which contain largely unstructured documents. Hypermedia is generally considered to provide very natural access to naive or casual users of small to medium scale document bases. However, the browsing access method becomes relatively clumsy and slow as the user becomes expert and as the size of the document base grows. Free text retrieval, on the other hand, naturally provides access to very large document bases and can be very efficient for expert users. However, query based retrieval does tend to have poorer results with novice users. Halasz (1988) presented a list of seven issues which he considered should be tackled by the next generation of hypermedia systems, amongst his list were search and query in a hypermedia network, and virtual structures for dealing with changing information. This section will present a model of combined hypermedia and free text retrieval and will then discuss the above issues with respect to such a system.

2.4.1 General Description

To provide access to a document base by either query or browsing, each node must have a descriptor and/or a set of links associated with it. This can easily be modelled as a variation of the previous document base descriptions for free text retrieval and hypermedia.

$\text{Base} = \{ N_i \mid i \in \mathbb{D} \}$	<i>all accessible nodes</i>
where	
$\mathbb{D} \subset \mathbb{Z}^+$	<i>currently valid node identifiers (integers)</i>
$N_i = \langle C_i, D_i, L_i \rangle$	<i>triple of content, descriptor, and links</i>
$D_i = \llbracket C_i \rrbracket$	<i>the indexing/evaluation of content C_i</i>
$L_i = \{ j \mid j \in \mathbb{D} \wedge N_i \rightarrow N_j \}$	<i>nodes which are linked to from N_i</i>
C_i	<i>the content of the node to be displayed</i>

This definition states that each node has associated with it a descriptor and/or a list of links, or possibly neither (since the list of links and the descriptor may be empty). To guarantee access to all nodes it must be shown that $\forall i \in \mathbb{D}: D_i \neq \emptyset \vee N_H \rightarrow N_i \vee (\exists j : D_j \neq \emptyset \wedge N_j \rightarrow N_i)$ where \emptyset is the empty descriptor (i.e. a descriptor which cannot match any query), and N_H is a home node from which a user may start browsing (if no such node exists then $N_H \rightarrow N_i$ is defined to be false). Although a more complex expression than was given for free text retrieval or hypermedia, it states a much weaker condition; access must be shown to every node by query ($D_i \neq \emptyset$), or by browsing from the home node ($N_H \rightarrow N_i$), or by following links from a node which can be accessed by query ($\exists j : D_j \neq \emptyset \wedge N_j \rightarrow N_i$).

Again this model does not express many of the details which are required to implement a system (e.g. the indexing and matching algorithms), it also takes a very simplistic view of hypermedia (e.g. it does not contain typed links or nodes). The model can be extended in the same fashion as described in the previous two sections for free text and hypermedia access. The discussion in this section does not rely upon any of these specifics and may be implemented using the very simple model or greatly enhanced models.

2.4.2 Search and Query in a Hypermedia Network

Halasz asserts that “effective access to information stored in a hypermedia network requires query-based access to complement navigation”. This statement, though true for all hypermedia networks, is especially relevant for very large networks in which it is unreasonable to expect users to successfully browse to the required information. Frisse (1988) presented the first work which fully merged query-based free text retrieval and hypermedia browsing (an update on the project was presented at the Hypertext ‘89 conference (Frissé 1989)). He states that queries in a hypermedia system should provide “one or more *optimal* starting points for graphical browsing”, a result of this approach is

that the matches to the query need not, in themselves, answer the query but may give access to areas of the document base which do answer the query. To achieve this Frisse states:

“the utility of a card [or node] can be approximated by a computed numeric weight consisting of two components. The *intrinsic* component is the value computed from the number and identity of the query terms contained in the card. The *extrinsic* component is the value computed from the weight of the immediate descendant cards.”

As a specific example, consider several sibling nodes which match the query, it may be better to retrieve the common parent rather than many individual nodes, since the user can browse from the parent node to any of the matched children (and to some children which did not match but are likely to be on the same topic). Frisse discusses work with a large hierarchical document. It is not clear how well the technique of retrieving the parent instead of multiple children would perform when used in a general graph structured hypermedia. The extrinsic component of a node would have to be defined as being computed from the intrinsic components of successor nodes (i.e. nodes which can be reached from the current node). The heuristics for what node should be retrieved instead of each specific node would have to be extended to cover the more general graph network, rather than a tree. An example heuristic could be: retrieve the start of a path if 50% of the elements in the path match the query. The model presented in chapter 3 could be used as a basic implementation of these ideas. In the later versions of this model all nodes are defined partly in terms of their neighbours, this does, however, only take mild account of neighbours and does not remove the neighbours from the matched document list. The problem of whether more general nodes (parents or neighbours) should be retrieved raises the issue of the style of query for which an individual system will be used. If the users are typically searching for a specific answer to a specific question (e.g. “where can you pick up hitch-hikers on a motorway”) then the nodes themselves should be retrieved, as the answer will probably be found within a single node. If the queries are typically more general, searching for general information about a topic (e.g. “driving on motorways”), then more general nodes should be retrieved as it is likely that the user wants to read a larger section of the document base. It may be possible to analyse the matched node list to establish which kind of search was performed. If many nodes have similar matching scores then it is likely the query was general, but if a few nodes have much higher weights than others then it is likely that the query was of a specific nature and the matching rules should be retrieved.

Similar work into combining queries with browsing based access has been carried out by Savoy and Desbois (1991), who have extended Frisse’s model of retrieval from a hypermedia document base, which generalises from the specific domains considered by Frisse, by use of Bayesian inference networks. They do not, however, fully handle the issue of the behaviour of this approach within a general graph rather than within a tree.

With a different emphasis between querying and browsing, Thompson and Croft (1989) describe a system that they developed, I³R, in which an information search is initiated by entering a textual query in free-form English prose. In response to this query the retrieval system presents a list of matched documents which the user may open to examine the content of (in the case study, abstracts from the CACM collection are used). While viewing a node the user can start browsing, this results in a map being displayed. This map has the current node at the centre and various *connected* nodes surrounding the centre which can be moved to by clicking with the mouse. Although the CACM collection provides some citation information, this information does not provide the majority of links directly, instead links are created to documents or concepts which are likely to be relevant to the user. Link types include links to the nearest neighbour of this document, to the concepts discussed in this document, and to cited documents – the system decides whether a link should be displayed on the basis of how likely the node is to be relevant to the user. Citation links, as well as the relevance-based links, are filtered by this process. I³R provides a very high level of integration between querying and browsing. However, since all links are to some extent calculated by the system in response to the current query, it cannot be described as a hypermedia system. The provision of only query-dependant links may strongly reduce the benefits of providing a true hybrid system in which the user can use querying to get within the *physical* neighbourhood of the required documents and then browse through this neighbourhood.

Thompson and Croft also developed a notion of user modelling within a retrieval system: when users log on to the system they are asked various questions, the answers to which are used to establish whether they are novice or expert users of the system, expert or novice in the field of interest, and whether an exhaustive or selective search is required. The decisions which are made after the initial *stereotyping* questions affect many areas of the user interface and should provide a system which is better tuned to the needs of the current user.

2.4.3 Virtual Structures for Changing Information

Thompson and Croft presented many methods for creating transient links as the results of a query in the I³R system, these have been discussed in the previous sub-section. Watters and Shepherd [1990] present work on displaying and manipulating the results of a database query as a transient structure (although they have also shown the approach to work in a free text retrieval system). A hypermedia-style graph is produced from the set of nodes which have matching attributes in the underlying database, and allows the users to browse the neighbourhood of their result set. As an example consider a user issuing a query for “all memos written by I. Cathcart dated yesterday”, from the results of this query the user would be able to browse to (security permissions permitting) other memos written yesterday, or to other memos written by I. Cathcart, or to other documents written by I. Cathcart and from their to items related to those documents. Watters and Shepherd

also describe their work on the New Oxford English Dictionary in which they have developed an access model to the nodes of the dictionary which is based on transient hypergraph techniques. The user initially enters a query which is matched against the entries in the dictionary. Matched entries are then outlined within a window from which the user can browse to any of the entries, and from those entries the user can start a new query by clicking on a term in an entry window. The new query results in a list of matched nodes which is also displayed on screen, in this way the user can browse throughout the document base which has no links (except between the use of a word and its definition).

Yankelovich and Meyrowitz (1985) and Garg (1988) suggest using an information retrieval-style engine to filter a hypermedia document base so that users are not overwhelmed with articles which are completely irrelevant. It would be possible for users to initially enter a textual query or to simply start browsing. Thereafter they could re-formulate their query (or formulate it) by providing relevance feedback. The retrieval engine could then be used to filter out nodes which almost certainly do not match the user's information requirement and leave users to browse through the remaining structure. This is very different from a conventional retrieval engine which must retrieve a small number of nodes which are expected to be relevant to the query. Here the retrieval engine is expected to filter out those nodes which are extremely unlikely to match the query. This approach develops another major issue recognised by Halasz (1988) as a requirement for the next generation of hypermedia systems, i.e. "queries can be used as a filtering mechanism in the hypermedia interface". Yankelovich and Meyrowitz also suggest that keywords can be associated with links. Though their discussion is aimed at using this information to aid users understand the purpose of links, the notion could be extended so that a link may be defined as a descriptor. In this case the results of following the link would be the retrieval of any nodes in the document base which match the descriptor. For example, a node concerning studying at Glasgow University may contain a link marked "local scenery" which, when activated, would search the current hypermedia network for any nodes which discuss Scottish scenery. A free text retrieval engine behind the scenes of a hypermedia system could also be used to provide content based links, for example, links to the current node's nearest neighbour.

An interesting approach is presented by Ducloy *et. al.* (1991) in the context of the KWICK Esprit project. They describe the building of document clusters (see section 2.2.8) for browsing purposes. In the system the document base is partitioned into clusters based on the content of the documents. These clusters are then structured so that the user can browse through a cluster hierarchy to reach documents which should be very similar. After reaching useful documents the neighbourhood can also be browsed. While not permitting the dynamic change of a document base structure, due to the complexity of the cluster calculations, this approach does provide for automatic re-building of the clusters in response to a changing document base.

Aigrain and Longueville (1991) present a system which allows access to a document base of images (the video disc “Images of the French Revolution”) by browsing through an automatically calculated structure. The textual descriptions of the images (taken from the *Bibliothèque Nationale* database) are used to create a graph which the user browses. The browsing technique is quite different to that normally found in hypermedia systems because the system is designed solely for the retrieval of images. The user is presented with eight small images on the screen, these images surround the image of the current position in the image base and are all similar, but in some sense different, to the current image. To move to the next image the user simply selects one of the similar images, this is then displayed as the current image and eight other images are displayed which are similar to this new image.

A hypertext interface to a traditional boolean retrieval engine is presented by Bovey and Brown (1987), who use the Guide hypertext system to format and present the results of queries. Although hypertext links are not provided between nodes, the approach does demonstrate how hypertext technology can be used to provide an intuitive browsing interface to a query based system. They make extensive use of *glossary* links, standard links where the destination is displayed within a new window, and *replacement* links, by which the source of a link is a string which when selected is replaced in line by the destination. A query results in a glossary link, the source of which states how many matches there were for the query. Users can then examine the list of matched nodes by following this link to a new window which lists the article titles. These titles are followed by an replacement link, ‘More’, which when expanded shows the additional field of the matched article together with more replacement buttons for further details. It is useful to compare this approach with that of the Justis project (Wilson 1988) which uses Guide in a very similar manner but purely for browsing (i.e. without the use of the query-based retrieval engine).

2.4.4 Access to Hybrid Document Bases

So far in this section we have considered approaches to combining the features of hypermedia and free text retrieval. There has, however, been little consideration of the general retrieval model through which users would access the system. Access can be provided through an entirely browsing environment, an entirely querying environment, or a combination of browsing and querying, the choice depends greatly on the size of the document base and the expected set of users.

If the document base is reasonably small and/or the users are not expected to be experts, a system which provides only browsing access may be preferable. With such an approach the free text retrieval engine could be used to provide extra links to the nearest neighbour of the current node, or indeed to other sections which may be of interest. This would reduce the categorising effect of hypermedia systems, by which users cannot find information because it is stored under a different category than they expected. This is

especially relevant when the document base has a strongly hierarchical structure. The user would still have to find a document similar to the one for which (s)he was looking. For example, when searching for information in the Highway Code on flashing amber lights at pelican crossings¹⁰, a user would not find this under “road user on foot /crossing the road/pelican crossings” because this section describes the pedestrians view point. If users were given access to a list of similar nodes then it would be possible to browse to the rules which discuss pelican crossings and hopefully find the answer even though it is held in “road user on wheels/driving along/safety of pedestrians”. Within a browsing-only interface it would be possible for the user to provide relevance feedback when discovering nodes which are of interest. This feedback could then be used to rank the list of similar to nodes or possibly to emphasise links which were to nodes which matched the current (feedback-based) query. This would allow the user to browse around the document base, and slowly, as relevance feedback is given, the links would be emphasised to help the user choose routes which are likely to be useful. If authors of hypermedia documents were given access to queries then they could use these to scan the document base for nodes which should be linked to their document. This would reduce the theoretical requirement for authors to scan the entire document base when deciding which other nodes should be linked to/from the new nodes.

If access to the document base is restricted to query only, for example if the retrieval system must interface to other systems or facilities, then the existence of hypermedia links are only useful for access to non-textual nodes. Chapter 3 describes a method whereby the descriptor of textual nodes which are linked to a non-textual node can be used to describe that node. With a query-only user interface there would be no links visible to the user, but some links could be created to provide access to the non-textual nodes by query.



In a system where the user is given access to both browsing and querying commands there is a choice of how the user should access the information. In large, weakly linked, document bases users would be required to initiate an information search with a query, this would then return a list of nodes which will either answer the users information need or will serve as starting points for browsing. However, if the document base is small enough for browsing-based access then it may be better to provide browsing as the primary form of access but allow users to enter queries when they cannot find the required information by browsing. In either case the benefits of calculated links and access to non-textual nodes by query are available.

2.5 MultiMedia Document Systems

Although traditional free text retrieval is not directly applicable to non-textual domains some systems have been developed which allow access to non-textual documents by query, using information retrieval or other techniques. A system developed by Harabayshi,

¹⁰ The drivers indication that (s)he should give way to pedestrians who are already crossing the road.

Matoba, and Kasahara (1988) uses traditional information retrieval techniques in an innovative manner to provide access to photographs of clothes held in a fashion document base. Each photograph is indexed according to various impressions which the outfit may have on users, example impressions include *formal*, *casual*, *bright*, and *elegant*. These impressions then form a vector of real values which can be matched against the users current request which is a similarly constructed vector. The process, although not capable of supporting a large document base on its own, could be used in conjunction with database-type queries to provide a powerful tool, e.g. give me all bright and formal skirts by European designers is an example of a mixed DRS/DBMS style query, in which skirt and European designer would be fields of matching database records with bright and formal matching the impression of the skirt. One major problem with this method is the subjectivity of impression-based descriptions, for example a student's concept of casual may be very different from a managing director's. Harabayshi *et al.* suggested that this problem could be removed by having the document base indexed by a single person, users would then automatically adjust their impressions relative to the document base's in a similar manner to the change of impression shoppers have when moving between different style of clothes shops. Although this adjustment is likely to occur, the success of such a system is limited by the consistency of the indexer. There may still be inconsistencies within the indexes created by a single indexer, for example when fashions change the definition of *bright* will also change. The approach of using a single indexer also raises issues of continuity when that person leaves their job and of the scale of the document base. A similar system was developed by Kato *et al.* (1991) to provide access to a museum's art collection by computer. They also developed a system for the automatic retrieval of icons (or logos). This later system relies upon the user issuing pictorial queries and the retrieval system responds with similar icons to the query drawing.

Two similar search routines were developed by Constantopoulos, Drakopoulos, and Yeorgaroudakis (1991) and by Kurlander and Bier (1988) for, respectively, a vector-based drawing document base and a vector-based drawing editor. Both algorithms exploit the underlying structure of the vector-based object to match with entries in the document base (or the drawing currently being edited), and they both provide various means for the user to reduce the precision of the matching to attempt to improve the search's recall. Constantopoulos *et al.* present a method for retrieving vector images from a document base after issuing a pictorial query. Queries are created by the user drawing what (s)he is interested in, the system then searches for similar images or for images which contain the query image. The model for query document matching which they present appears to be quite flexible, although the examples given are of very precise matches, so it is not clear how well the system performs with poorly matching queries. Kulander and Bier present a vector drawing system which allows the user to draw an image to search for, and one to change the image to. Examples in the paper include many uses of the method to manually create fractal images (e.g. by repeatedly replacing  with ). They also allow users to

search for various properties of objects (e.g. colour, line width, text font) to change as well as the entire object. Users can also change object shape but maintain the other attributes. Both these systems are, however, subject to retrieval failures imposed by the underlying structure which is used to compose an image. There are many circumstances in a vector-based drawing system in which a single object can be constructed in radically different ways such that one of these constructions will not match another. As an example consider the construction of a semi-circle, most drawing programs do not provide these as primitive objects so the user must construct them, there are two sensible approaches: draw a full circle and place a rectangle of the background colour over the circle, or draw an arc of 180° . Although the second approach may be slightly more natural, the first is still a sensible way of constructing a semi-circle. When using a matching algorithm based on the structure of drawings, these two visually identical objects could not be expected to match – except by the inclusion of a special case or a visual thesaurus.

Rabitti and Savino (1991) present a similar system which uses a rule base to automatically describe objects within a picture with respect to a set of known objects. These objects may be stored in vector format or in bitmap format. However, they must be reasonably simple, well structured and two dimensional. The image analysis first tries to find members of the set of known objects within the image. It then attempts to use a rule base to recognise complex, or compound, objects. In the paper an example is given of house floor plans with basic objects as pieces of furniture, walls, etc. Rules can be expressed to state that, for example, “a dining room must contain a dining table, a buffet is optional but important, a sofa is optional, while objects like bidet, lavabo, etc. are highly improbable.” The term *dining table* can, itself, be defined as a “table, with at least two chairs and less than 12 chairs. The chairs must be *CLOSE* to the table.” While this approach requires the system to be taught in advance the set of basic objects and the rule set, it does provide a powerful model for domain specific document bases in which there is a basic set of objects which can be drawn iconically as 2-D images. The images can be retrieved by textual queries on the relationships between objects (simple and complex) within the image base.

The Picture Archive System being developed at the University of Singapore (Al-Hawamdeh *et. al.* 1991a, 1991b) uses textual descriptions which are associated with images to give the impression of automatically retrieving images and of providing relevance feedback on images. The system presents the result of a query as a set of small images within a single window, from this display of small images the user can either give relevance feedback, zoom the image to display it full size, or examine the text which is associated with the image. The use of greatly reduced versions of the images to show the matched document list within one window provides a very rapid method for users to select images which they consider relevant which is not available to textual document bases (in which summary information must be given, e.g. title and author). A somewhat similar system is described by Halin, Créhange, and Kerekes (1990) who describe how machine-

-learning can be used to retrieve images based on user feedback. They conclude by stating that “retrieving images needs a deep interactivity and cooperation between man and machine and, thus, that the retrieval process may be viewed as a machine-learning process”. It is likely that the benefits of relevance feedback will be more apparent in the retrieval of non-textual documents than for textual documents.

2.6 Context of Thesis

The research reported in the remainder of this thesis is based around providing access to non-textual nodes by query. To achieve this, a document base which provides access to nodes by query or by browsing (as described in section 2.4) is used. Relevance feedback (see section 2.2.4) within such a document base is discussed in chapter 5. This discussion concentrates on the effect of users being able to give relevance feedback on documents which do not match the current query. A retrieval system, *mmIR*, is described in chapter 6. *mmIR* provides access to the British Highway Code by querying, browsing, or a combination. It was mainly developed to access the model of accessing non-textual nodes described in chapter 3, and to assess the various benefits of each access method to a document base which can support all methods. User tests were ran to assess the various access methods to *mmIR*, and to consider other user interface issues. These tests are described in chapter 7.

2.7 Summary

This chapter has given an overview of work which has been carried out within the traditional fields of free text retrieval and hypermedia (and hypertext). Approaches to the combination of querying and browsing have also been discussed, and this combination appears natural and beneficial since the benefits, and drawbacks, of both access methods appear to be orthogonal. Hypermedia performs very well for casual or naive users, and within small document bases or small sub-sections of a document base; it also provides excellent access to non-textual information. However, hypermedia-based access does become relatively slow and inefficient as users become expert, and as the size of the document base grows, and the authoring of links becomes almost impossible with very large document bases (i.e. millions on nodes). Query-based retrieval has almost an opposite set of properties: searches can easily be performed on very large document bases, expert users can use queries efficiently, authors need not concern themselves with the size or structure of the document base, but access to non-textual documents is very restrictive and performance of novice users can be poor.

The combination of query and browsing based access which was first suggested by Frisse (1988) provides a powerful combined access method. With combined access, queries can be used to locate areas of the document base which are of interest and then browsing techniques can be used to examine these sub-sections of the hypermedia network. The combined model can also be used to access non-textual nodes held within

the document base: either by browsing from textual nodes which matched the query or by using the context based techniques described in chapter 3. Access to non-textual documents by query on content has been restricted to very specialised domains or to specialised vector based drawing systems.

3 A Model of Access to Non-Textual Nodes

3.1 Introduction

This chapter describes the underlying model for access to non-textual nodes by considering the neighbourhood (or context) of the node in the document base. The chapter starts by formalising the traditional method which does not provide query-based access to non-textual nodes, but provides only access through browsing. The chapter then goes on to describe various extensions of this model which provide increasingly competent node descriptors. The concept of defining a document's content, or style, in terms of other documents is not new and is used extensively in the fine art domain, for example the work of a new painter may be described as "similar to the later work of De Kooning but doesn't have the same vibrancy, in this sense it is more like early Robert Motherwell".

This chapter shall give definitions for node descriptor calculations in general, and will also give specific definitions for the nodes which are held in the network shown in figure 3.1. Within the main text of this chapter only some examples will be given from the sample network. In a chapter appendix (section 3.9) the full definitions are given for every node in the network. This network is composed of six nodes: four textual (C_0 , C_3 , C_4 , and C_5) and two non-textual (C_1 and C_2).

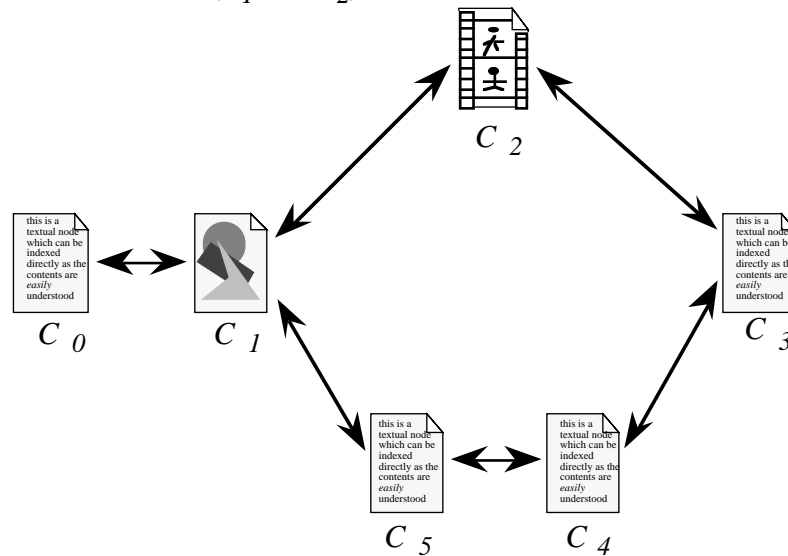


Figure 3.1 Sample document base

The chapter first presents the traditional model of retrieval from multimedia document bases in which non-textual nodes cannot be retrieved by query. This model is presented to introduce the notation and formalise the simple approach. The chapter then goes on to describe the model of retrieval used in the prototype application (*mmIR*), and various extensions to this model. Initially the descriptions shall be based around a very simple model of hypermedia in which all links are considered as bi-directional (i.e. all links can

be followed in either direction), are of a single type, and are not weighted. Towards the end of the chapter a model is described which will add these factors.

3.2 Simple Approach

The traditional model of hybrid access (Frisse 1988) is based on the assumption that users can gain access to nodes based upon their node descriptors. Frisse does not consider non-textual nodes, but under his model users would have to browse to non-textual nodes from their textual neighbours. This approach to descriptors can be formalised as below, throughout this thesis double square brackets are used to define the operation of creating a node descriptor from a given node's content (for example $D = \llbracket C \rrbracket$ states that descriptor D is set to the index calculated from the content C). When it is not possible to index a node's content the value can be considered as empty (or null), and is represented by \varnothing . The simple approach can be defined, in general, as:

$$\begin{aligned} D_i &= \llbracket C_i \rrbracket && \text{if } C_i \text{ is textual} \\ &= \varnothing && \text{if } C_i \text{ is non-textual} \end{aligned}$$

with examples in the sample network defined as:

$$\begin{aligned} D_0 &= \llbracket C_0 \rrbracket \\ D_1 &= \varnothing \end{aligned}$$

where C_i and D_i are respectively the node content and descriptor of node i .

The above model provides access by query to textual nodes, however, access to non-textual nodes is not possible directly through queries but only through browsing.

3.3 Simple Cluster Approach

The set of textual nodes neighbouring a non-textual node could be considered as a cluster of nodes giving the context of that non-textual node. A descriptor for the non-textual nodes can then be calculated based on the descriptors for nodes in this cluster. One method of achieving this context-based descriptor is to use the *cluster representative* of the set of neighbours. Cluster representatives are used in information retrieval to summarise the content of a cluster and thus require the retrieval engine to access only the representatives and not every member of the cluster (van Rijsbergen 1979 chp. 3, Salton 1971, Croft 1978). A commonly used cluster representative is the *cluster centroid*, this method models the cluster as a set of points in N dimensional space (where N is the total number of terms used in all index files) and calculates the point in space which is at the centre of the points in the cluster. This can be implemented simply by taking the average

of the descriptors, or by summing the descriptors in the cluster and then normalising the result (so that it has unit length).

This model is used for the implementation of *mmIR*, described in chapter 6, and for the validation experiment described in chapter 4. Within these definitions the symbol \oplus is used to denote vector addition and the function $norm(\mathbf{V})$ is defined as the euclidean normalisation of the vector \mathbf{V} , i.e. $norm(\mathbf{V}) = \mathbf{V} / \|\mathbf{V}\|$ where $\|\mathbf{V}\|$ is defined as $\sqrt{\sum_{i=1}^N V_i^2}$ and N is the dimensionality of \mathbf{V} . The normalisation function results in a vector which has unit length, i.e. $\|norm(\mathbf{V})\|=1$. The simple cluster approach is defined as:

$$\begin{aligned} D_i &= \|\mathbf{C}_i\| && \text{if } \mathbf{C}_i \text{ is textual} \\ D_i &= norm\left(\bigoplus_{j \in N} \|\mathbf{C}_j\|\right) && \text{if } \mathbf{C}_i \text{ is non-textual} \end{aligned}$$

where N is the set of textual nodes which neighbour \mathbf{C}_i . Examples from the sample network are:

$$\begin{aligned} D_0 &= \|\mathbf{C}_0\| \\ D_1 &= norm(\|\mathbf{C}_0\| \oplus \|\mathbf{C}_5\|) \end{aligned}$$

This method allows access to all media by direct query: textual nodes are accessed by their content and non-textual nodes are accessed by their context in the hypermedia network.

3.4 Extended Cluster Approaches

The approach shown above provides access to textual nodes by content and to non-textual nodes by context. This produces an asymmetry in the treatment of different types of nodes. If we consider the evaluation of a descriptor based on the descriptors of its neighbours then we create the following set of recursive functions. These definitions are based on taking a weighted sum of the node's own content with the descriptors of the node's neighbours.

$$D_i = norm\left(k\|\mathbf{C}_i\| \oplus \bigoplus_{j \in N} D_j\right)$$

where

$N = \{ x \mid \langle x, i \rangle \in L \vee \langle i, x \rangle \in L \}$ neighbours of node i in the network

$L = \text{set of pairs representing all links in the document base}$

$k = \text{constant greater than 1 defining how important the node's content is with respect to its context}$

from the sample network:

$$\begin{aligned} D_0 &= \text{norm} (k \llbracket C_0 \rrbracket \oplus D_1) \\ D_1 &= \text{norm} (k \llbracket C_1 \rrbracket \oplus D_0 \oplus D_2 \oplus D_5) \end{aligned}$$

The constant k provides a method of stating how much a node's descriptor should be defined in terms of its own content and of its context (or neighbouring nodes). One can consider k as stating how many of the neighbouring descriptors are equivalent to a node's own (indexed) descriptor. These definitions rely on fully defined descriptors of neighbouring nodes and, therefore, are deeply recursive. These definitions can, however, be unfolded one level at a time to produce increasingly accurate approximations to the full definitions.

3.4.1 Level 1 Cut Off

Initially we can consider approximating each D_i on the right-hand side of the equations to be equal to $\llbracket C_i \rrbracket$, this results in the following definitions for the examples from the test network (see figure 3.1):

$$\begin{aligned} D_0 &= \text{norm} (k \llbracket C_0 \rrbracket \oplus \varphi) \\ &= \text{norm} (k \llbracket C_0 \rrbracket) \\ D_1 &= \text{norm} (k\varphi \oplus \llbracket C_0 \rrbracket \oplus \varphi \oplus \llbracket C_5 \rrbracket) \\ &= \text{norm} (\llbracket C_0 \rrbracket \oplus \llbracket C_5 \rrbracket) \end{aligned}$$

and in general:

$$D_i = \text{norm} \left(k \llbracket C_i \rrbracket \oplus \bigoplus_{j \in N} \llbracket C_j \rrbracket \right)$$

This provides a first approximation to the recursive definitions and a method for describing the content's of any node based on a combination of its content and of its neighbours'. The definitions which are specific to the test document base show that for non-textual nodes these definitions are the same as those described in section 3.3, but textual nodes are now also (partly) defined in terms of their neighbours.

3.4.2 Level 2 Cut Off

The recursive descriptions given at the start of section 3.4 can be unfolded once to provide the following expressions (for the examples from the sample network):

$$\begin{aligned} D_0 &= \text{norm} (k^2 \llbracket C_0 \rrbracket \oplus k \llbracket C_1 \rrbracket \oplus D_0 \oplus D_2 \oplus D_5) \\ D_1 &= \text{norm} (k^2 \llbracket C_1 \rrbracket \oplus k \llbracket C_0 \rrbracket \oplus D_1 \oplus k \llbracket C_2 \rrbracket \oplus D_1 \oplus D_3 \oplus k \llbracket C_5 \rrbracket \\ &\quad \oplus D_1 \oplus D_4) \end{aligned}$$

Here the term k^2 is used to take into account the normalisation process which takes place when a descriptor is assigned to a node. Any remaining D_i on the right-hand side of these expressions can now be approximated by $\mathbb{I} C_i \mathbb{I}$ producing the following equations:

$$\begin{aligned} D_0 &= \text{norm} (k^2 \mathbb{I} C_0 \mathbb{I} \oplus k \mathbb{I} C_1 \mathbb{I} \oplus \mathbb{I} C_0 \mathbb{I} \oplus \mathbb{I} C_2 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I}) \\ D_1 &= \text{norm} (k^2 \mathbb{I} C_1 \mathbb{I} \oplus k \mathbb{I} C_0 \mathbb{I} \oplus 3 \mathbb{I} C_1 \mathbb{I} \oplus k \mathbb{I} C_2 \mathbb{I} \oplus \mathbb{I} C_3 \mathbb{I} \\ &\quad \oplus k \mathbb{I} C_5 \mathbb{I} \oplus \mathbb{I} C_4 \mathbb{I}) \end{aligned}$$

By removing zero descriptors, i.e. indexed descriptors for non-textual nodes, these equations further simplify to the following:

$$\begin{aligned} D_0 &= \text{norm} ((k^2+1) \mathbb{I} C_0 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I}) \\ D_1 &= \text{norm} (k \mathbb{I} C_0 \mathbb{I} \oplus k \mathbb{I} C_5 \mathbb{I} \oplus \mathbb{I} C_3 \mathbb{I} \oplus \mathbb{I} C_4 \mathbb{I}) \end{aligned}$$

In general, level 2 cut off is defined as:

$$D_i = \text{norm} \left(k^2 \mathbb{I} C_i \mathbb{I} \oplus k \bigoplus_{j \in N} \mathbb{I} C_j \mathbb{I} \oplus \bigoplus_{j \in M} \mathbb{I} C_j \mathbb{I} \right)$$

where

$$N = \{ x \in B \mid \langle x, i \rangle \in L \vee \langle i, x \rangle \in L \} \quad \text{all immediate neighbours of node } i$$

$$M = \{ x \in B \mid \exists y \in B : (\langle x, y \rangle \in L \wedge \langle y, i \rangle \in L) \vee (\langle i, y \rangle \in L \wedge \langle y, x \rangle \in L) \}$$

all neighbours of node i with exactly one
intervening node (including return paths)

These expressions define the descriptor of a node in terms of its immediate neighbours and also in terms of its immediate neighbours' immediate neighbours – that is, all nodes which can be reached by following exactly one or two links. This extends the set of network configurations which are supported by the cluster-based descriptor method: previously a non-textual node had to be directly connected to textual nodes to be accessed by query. With level 2 cut off, any non-textual node, which can be reached by at most two links from a textual node, can have a descriptor assigned (and therefore be accessible by query).

Level two cut off is considerably harder to calculate than level one (or indeed the basic cluster approach described in section 3.3). In general, for a document base of N nodes, with an average of M links per node, the amount of work involved in calculating these descriptors for level L cut off can be stated as $O(N.M^L)$. The $O(e)$ notation states that the complexity rises no faster than that of the expression e , the definition given here is only valid for positive integer values of L .

Level two cut off provides a generous contribution to the descriptor of a node from its neighbours and may significantly improve descriptors for poorly connected non-textual

nodes. It is, however, expensive at index time to calculate the descriptions (an $O(N.M^2)$ algorithm as opposed to $O(N.M)$ for level 1 cut off). It is unlikely that an extension to a level 3 cut off would provide any great improvement in retrieval performance since the additional nodes would be quite distant from the node being described.

3.5 Consideration of Type Information

In section 3.4 the algorithms described all use a constant k to control the balance of a node's own content with the descriptors of its neighbours. An extra level of control could be achieved by introducing a function, f , which would take various attributes into account to decide the importance of that descriptor for the current calculation. The function f is defined here with its first parameter as the node number of the descriptor that is being calculated, the second and third parameter are respectively the source and destination of the link being considered. This would result in the general equations for level 1 and level 2 cut off being re-written as below.

$$\text{level 1: } D_i = k.f(i,i,i) \| C_i \| \oplus \bigoplus_{j \in N} f(i,i,j) \| C_j \|$$

$$\text{level 2: } D_i = (k.f(i,i,i))^2 \| C_i \| \oplus k \bigoplus_{j \in N} f(i,i,j) \| C_j \| \oplus \bigoplus_{\langle x,y \rangle \in M} f(i,x,y) \| C_x \|$$

where

$$N = \{ x \in B \mid \langle x,i \rangle \in L \vee \langle i,x \rangle \in L \}$$

$$M = \{ \langle x,y \rangle \in B \mid \exists y \in B : (\langle x,y \rangle \in L \wedge \langle y,i \rangle \in L) \vee (\langle i,y \rangle \in L \wedge \langle y,x \rangle \in L) \}$$

$$f(i,j,k) \in [0..1]$$

If, for example, character recognition software was used to extract any textual content from a raster image, D , then its evaluation, $\| D \|$, would include the encoding of that text. This text would, however, be less suitable for calculating a descriptor than the text extracted from a textual node. This situation could be reflected by returning a lower value of f for non-textual nodes than for textual nodes. This could be further refined to take account of various types of raster image: for example, general photographs, application forms, or scanned textual documents. With this model it would also be possible to remove the effect of standard links (e.g. links to home or system help nodes) from the calculation of a node's descriptor.

Most hypermedia systems model links as directional. In such systems information concerning the direction of links may be taken into account when calculating the descriptor for non-textual nodes. The models described above, which were defined with bi-directional links in mind, do not explicitly address the issue of uni-directional links. There are three approaches to handling such links:

- Ignore links in one direction
- Consider links from and to the node as equals

- Weight one direction of links over the other

It is likely to be disadvantageous to completely ignore some links. However, it is not clear which direction of links should be weighted higher. In a hierarchical system the links into a node will be from more general nodes, whereas the links out of a node will be to more specific nodes. Within a general hypermedia graph this is not the case. It is also not clear which direction of links would be more important. Within an individual document base it may be beneficial to introduce weighting on the direction of link, but in general links into a node should be considered as strongly as links out of a node.

Taking account of the link type may also be beneficial as many hypermedia systems provide different types of links which are, presumably, not all of equal strength. As an example, when evaluating a descriptor for a television programme stored in an entertainment document base, the programme may have several textual nodes associated with it. The programme's script should be considered as a very important link, since it is almost a textual equivalent of the programme. Reviews in the press may also be considered with reasonable strength, but technical notes relating to the programme may be rated lower. It is also possible that hypermedia models could provide the author with a method for weighting links so that the user has an impression of how strong the author thought the connection was. Such link weights could be used directly by the function f to establish the importance of links.

In general the inclusion of more information into the process of descriptor calculation should lead to an improvement in the quality of descriptors.

3.6 Mixed Descriptor Types

The model of free text retrieval which is used in *mmIR* and which is mainly discussed in this thesis, is suitable for use under many different indexing methods. Since the model represents all files as vectors of real values the model need not be restricted to working with term vectors. In a multimedia environment the vector could represent many attributes of an image, or other non-textual object, instead of terms in a traditional textual systems.

The impression-based retrieval system developed by Harabayshi, Matoba, and Kasahara (1988) for a fashion document base, is very amenable to a vector representation. Their document base is composed of photographs of fashion clothing, and is indexed by considering each node as a set of values representing various attributes of the item of clothing in the image. These attributes include some fields which are based on impressionistic descriptions of the clothing; example impression values are *casualness*, *brilliance*, and *elegance*. These attributes are given a numeric value which states, for example, how casual the jacket is. These attributes could easily be modelled as dimensions in N dimensional space and standard vector based retrieval methods could be used for these images, although Hirabayshi *et al.* use an alternative model of retrieval.

These approaches work very well in a single medium document base. However, it is envisaged that most document bases built from the model presented in this thesis shall be

composed of many media. These models can, however, be extended to support many different types of node descriptors (or vectors). The methods described earlier in this chapter allow the descriptors of textual nodes to *spread* to their neighbours by cluster based techniques. Identical approaches can be taken for non-textual vector-based descriptors. This thesis is proposing that non-textual nodes may be retrieved by using the textual descriptors of neighbouring textual nodes. It is also possible for textual nodes to be retrieved on the bases of their neighbours. This results in a document base which can be accessed through different query languages, possibly one query language per media, and that these queries will retrieve some items which cannot be directly indexed by that method, as well as those that can.

The underlying document base model would have to be changed to accommodate many different vector-based descriptors per node. A node can now be described as being composed of a content, C_i , together with an M-tuple of descriptors, D_i . The retrieval process and all the clustering techniques described above can be applied to any of the elements of the descriptor tuple.

3.6.1 A Multi-Descriptor Example

As an example consider the sample network shown in figure 3.1 and consider that all non-textual nodes are indexed by impression whereas all textual nodes are indexed by terms extracted from their contents. Using level 1 cut off (see section 3.4.1) the equations below are derived, where T_i refers to the text-based descriptor of node i , and I_i refers to the impression based descriptor. For the sample network:

$$\begin{aligned}
 T_o &= \text{norm} (k \llbracket C_o \rrbracket_1 \oplus \phi) \\
 &= \text{norm} (k \llbracket C_o \rrbracket_1) \\
 I_o &= \text{norm} (k\phi \oplus \llbracket C_1 \rrbracket_2) \\
 &= \text{norm} (k \llbracket C_1 \rrbracket_2) \\
 \\
 T_1 &= \text{norm} (k\phi \oplus \llbracket C_o \rrbracket_1 \oplus \phi \oplus \llbracket C_5 \rrbracket_1) \\
 &= \text{norm} (\llbracket C_o \rrbracket_1 \oplus \llbracket C_5 \rrbracket_1) \\
 I_1 &= \text{norm} (k \llbracket C_1 \rrbracket_2 \oplus \phi \oplus \llbracket C_2 \rrbracket_2 \oplus \phi) \\
 &= \text{norm} (k \llbracket C_1 \rrbracket_2 \oplus \llbracket C_2 \rrbracket_2)
 \end{aligned}$$

and in general:

$$T_i = k \mathbb{I} C_i \mathbb{I}_1 \oplus \bigoplus_{\forall j \in N} \mathbb{I} C_j \mathbb{I}_1$$

$$I_i = k \mathbb{I} C_i \mathbb{I}_2 \oplus \bigoplus_{\forall j \in N} \mathbb{I} C_j \mathbb{I}_2$$

where

$N = \{ x \in B \mid \langle x, i \rangle \in L \vee \langle i, x \rangle \in L \}$ the neighbours of node i

$D_i = \langle T_i, I_i \rangle$ the descriptor tuple

$\mathbb{I} C_i \mathbb{I}_1$ = the indexing of content C_i by textual content

$\mathbb{I} C_i \mathbb{I}_2$ = the indexing of content C_i by impression

This model provides us with a powerful method for retrieving documents held on a mixed-media document base. It allows various media to be described by media-specific techniques and for these descriptions to be shared with neighbouring nodes. The following sub-sections describe how such a model of retrieval could be used in practice.

3.6.2 Querying

To query a document base which is composed of mixed descriptors the user would have to supply a query in one or more of the descriptor media. The retrieval system could then rank the results (using standard techniques) and could merge the results of each media's query (if the user issued a query with mixed descriptor media).

The merge routine, which takes the results of several queries and combines them into one list of matched nodes, would have to take into account the size of the queries before assigning final ranks. This would be required to prevent the results being biased towards descriptor media in which the user's query is very poorly expressed: for example, if the user spent some time developing an impression query to a fashion document base and then enters a single word as a textual query one would expect the system to give greater weight to the impression query. The exact merge algorithm would have to be derived theoretically (or possibly experimentally) so that it produces the most suitable ranking order. It is not immediately clear whether the order should produce an outcome closer to a filter or to a merge operation. In the above example, if the single word was "skirt", one would consider this to be a filter operation (i.e. only retrieve skirts), whereas if it were "colourful" one would consider this as a merge with the impression-based description. It may be possible to determine the suitability of each method of combining the matched document lists by considering the content of the lists. If there are many documents which occur in both lists with similar matching values, a merge may be more suitable than a filter operation.

3.6.3 Relevance Feedback

Relevance feedback could be used to build up the users query in many different descriptor media. If the user marks a node as relevant then this should be taken into account on the next retrieval by all of the retrieval engines.

Typically a user would start a query in one descriptor medium and then browse around the document base giving relevance feedback. This feedback would be made on all descriptor media and so the next query that the user performs will be on the bases of many media - not just the media in which the original query was made.

3.6.4 Suitable Descriptors

Any model of retrieval which is based on vectors can be used for retrieval in hybrid document bases – either as the only model or as one of the descriptor types. Suitable models of retrieval include any model which describes the content of documents in terms of small primitives (e.g. terms or basic pictorial objects) and bases the retrieval upon the existence of such primitives.

Boolean vectors would also provide suitable descriptors for this model of cluster based retrieval. However, boolean vectors are not as expressive as real number vectors and this may lead to poorer quality clusters and, therefore, poorer quality of cluster based retrieval.

3.7 Limitations of Model

Although the model developed in this chapter provides a general purpose method of indexing non-textual nodes for access by textual query it does have some limitations. These are mainly connected with the quality and quantity of links which are available in a given document base. The model assumes that the document base contains many nodes which can have descriptors assigned from their content (in a traditional document base this set is restricted to textual nodes). This restricts use of the model to document bases which have a reasonable ratio of indexable (e.g. textual) to non-indexable nodes. In order for the cluster based-algorithms to provide some benefit, each non-indexable node should be linked to at least two indexable nodes. If only one link is available then the model degrades to a poor variant of the traditional method, by which non-indexable media are retrieved by indexing a hidden textual description. This restriction does not, however, require the network to be composed of twice the number of indexable nodes than non-indexable nodes. As an absolute minimum requirement, for level 1 cut off, each non-indexable node in the network must be connected directly to at least one indexable node. If the document base uses level 2 cut off, then the requirement is reduced to state that every non-indexable node must be connected, by no more than two links, to an indexable node. The precise ratio of indexable to non-indexable nodes which is required for reasonable cluster descriptors to be created is not clear, and will vary between document bases. In general a ratio of 1:1 could be considered as a reasonable minimum number of

textual nodes. As with many areas of free text information retrieval, there is no solid cut off point and the model can be used with lower indexable to non-indexable ratios, but the effectiveness of the retrieval engine will decrease with the percentage of non-indexable nodes. Likewise, the model benefits from increased ratio of indexable to non-indexable nodes and an increased number of links – so long as the number of links from each node is still a small subset of the document base. As the number of nodes which are used to calculate a cluster description approaches the total number of nodes in the document base, the effectiveness of the retrieval engine also reduces: as the cluster descriptor becomes a descriptor for the entire document base its ability to distinguish documents within the base is reduced.

As with plain hypermedia, the quality of links is also very important. The links must connect the node being indexed to other nodes which are similar. If the links are to very similar nodes then the descriptors will more precisely describe the content of the cluster and therefore more accurately describe the non-textual node.

3.8 Conclusions

This chapter has developed a model of information retrieval based on a hybrid document base of hypermedia structure with free text queries. The model provides access through cluster based descriptors to all media in the document base.

Various degrees of complexity are presented for the calculation of descriptors based on their context and these should provide suitable methods for a wide range of mixed media hypermedia document bases.

3.9 Full Definitions for Sample Document Base

This sections gives full descriptions for the sample network shown in figure 1 (repeated here for convenience). These definitions were omitted from the main chapter for simplicity but are included here for completeness. Throughout this section the general definitions will be given as D_i , while definitions from the test document base are given as D_0, D_1, \dots, D_5 . The textual descriptions of each set of equations are minimal within this section, the reader is referred to the main sections of this chapter for further details.

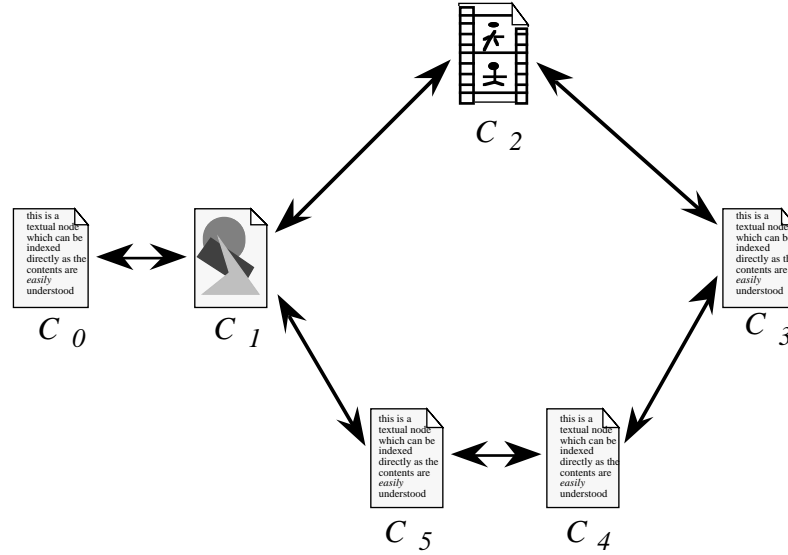


Figure 3.1 Sample document base

3.9.1 Simple Approach

The traditional, or simple, model of information retrieval from a multi-media document base only permits access to textual nodes by query. This can be defined as follows:

$$\begin{aligned}
 D_i &= \llbracket C_i \rrbracket && \text{if } C_i \text{ is textual} \\
 &= \varnothing && \text{if } C_i \text{ is non-textual}
 \end{aligned}$$

$$\begin{aligned}
 D_0 &= \llbracket C_0 \rrbracket \\
 D_1 &= \varnothing \\
 D_2 &= \varnothing \\
 D_3 &= \llbracket C_3 \rrbracket \\
 D_4 &= \llbracket C_4 \rrbracket \\
 D_5 &= \llbracket C_5 \rrbracket
 \end{aligned}$$

These equations define the descriptor of a textual node in terms of its content, but define the descriptor of a non-textual node as empty (\varnothing).

3.9.2 Basic Cluster Approach

The basic cluster approach, as used in *mmIR*, describes all textual nodes in terms of their content, and all non-textual nodes in terms of their neighbours' content. This can be expressed as follows:

$$\begin{aligned}
D_i &= \llbracket C_i \rrbracket && \text{if } C_i \text{ is textual} \\
D_i &= \text{norm} \left(\bigoplus_{j \in N} \llbracket C_j \rrbracket \right) && \text{if } C_i \text{ is non-textual}
\end{aligned}$$

where N is the set of textual nodes which neighbour C_i .

$$\begin{aligned}
D_0 &= \llbracket C_0 \rrbracket \\
D_1 &= \text{norm}(\llbracket C_0 \rrbracket \oplus \llbracket C_5 \rrbracket) \\
D_2 &= \llbracket C_3 \rrbracket \\
D_3 &= \llbracket C_3 \rrbracket \\
D_4 &= \llbracket C_4 \rrbracket \\
D_5 &= \llbracket C_5 \rrbracket
\end{aligned}$$

3.9.3 Extended Cluster Approaches

To define the two models of extended cluster description, a general recursive definition of descriptors was defined. This definition describes all nodes in terms of their own content, and their neighbours' descriptors. The general recursive definitions are as follows:

$$\begin{aligned}
D_i &= \text{norm} \left(k \llbracket C_i \rrbracket \oplus \bigoplus_{j \in N} D_j \right) \\
D_0 &= \text{norm} (k \llbracket C_0 \rrbracket \oplus D_1) \\
D_1 &= \text{norm} (k \llbracket C_1 \rrbracket \oplus D_0 \oplus D_2 \oplus D_5) \\
D_2 &= \text{norm} (k \llbracket C_2 \rrbracket \oplus D_1 \oplus D_3) \\
D_3 &= \text{norm} (k \llbracket C_3 \rrbracket \oplus D_2 \oplus D_4) \\
D_4 &= \text{norm} (k \llbracket C_4 \rrbracket \oplus D_3 \oplus D_5) \\
D_5 &= \text{norm} (k \llbracket C_5 \rrbracket \oplus D_1 \oplus D_4)
\end{aligned}$$

where

$N = \{ x \mid \langle x, i \rangle \in L \vee \langle i, x \rangle \in L \}$ neighbours of node i in the network

L = set of pairs representing all links in the document base

k = constant greater than 1 defining how important the node's content is with respect to its context

Level 1 Cut Off

As a first approximation to the recursive definitions all D_i on the right-hand side of the equations can be substituted with a calculation of the document's content, C_i . This leads to the following set of definitions:

$$\begin{aligned}
D_i &= \text{norm} \left(k \mathbb{I} C_i \mathbb{I} \oplus \bigoplus_{j \in N} \mathbb{I} C_j \mathbb{I} \right) \\
D_0 &= \text{norm} (k \mathbb{I} C_0 \mathbb{I}) \\
D_1 &= \text{norm} (\mathbb{I} C_0 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I}) \\
D_2 &= \text{norm} (\mathbb{I} C_3 \mathbb{I}) \\
&= \mathbb{I} C_3 \mathbb{I} \\
D_3 &= \text{norm} (k \mathbb{I} C_3 \mathbb{I} \oplus \mathbb{I} C_4 \mathbb{I}) \\
D_4 &= \text{norm} (k \mathbb{I} C_4 \mathbb{I} \oplus \mathbb{I} C_3 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I}) \\
D_5 &= \text{norm} (k \mathbb{I} C_5 \mathbb{I} \oplus \mathbb{I} C_4 \mathbb{I})
\end{aligned}$$

where

$N = \{ x \mid \langle x, i \rangle \in L \vee \langle i, x \rangle \in L \}$ neighbours of node i in the network

$L = \text{set of pairs representing all links in the document base}$

$k = \text{constant greater than 1 defining how important the node's content is with respect to its context}$

These definitions provide a consistent approach to the calculation of descriptors for all media. Every node in the document base has its descriptor calculated partly in terms of its own content, and partly in terms of the content of its neighbours.

Level 2 Cut Off

The general recursive definitions given at the start of this sub-section can be extended by substituting each occurrence of D_i in the right-hand side of an equation with its definition. This results in the following set of equations which are equivalent to those given at the start of this sub-section, but have been un-folded once.

$$\begin{aligned}
D_0 &= \text{norm} (k^2 \mathbb{I} C_0 \mathbb{I} \oplus k \mathbb{I} C_1 \mathbb{I} \oplus D_0 \oplus D_2 \oplus D_5) \\
D_1 &= \text{norm} (k^2 \mathbb{I} C_1 \mathbb{I} \oplus k \mathbb{I} C_0 \mathbb{I} \oplus D_1 \oplus k \mathbb{I} C_2 \mathbb{I} \oplus D_1 \oplus D_3 \oplus k \mathbb{I} C_5 \mathbb{I} \\
&\quad \oplus D_1 \oplus D_4) \\
D_2 &= \text{norm} (k^2 \mathbb{I} C_2 \mathbb{I} \oplus k \mathbb{I} C_1 \mathbb{I} \oplus D_0 \oplus D_2 \oplus D_5 \oplus k \mathbb{I} C_3 \mathbb{I} \oplus D_2 \oplus D_4 \\
&\quad) \\
D_3 &= \text{norm} (k^2 \mathbb{I} C_3 \mathbb{I} \oplus k \mathbb{I} C_2 \mathbb{I} \oplus D_1 \oplus D_3 \oplus k \mathbb{I} C_4 \mathbb{I} \oplus D_3 \oplus D_5) \\
D_4 &= \text{norm} (k^2 \mathbb{I} C_4 \mathbb{I} \oplus k \mathbb{I} C_3 \mathbb{I} \oplus D_2 \oplus D_4 \oplus k \mathbb{I} C_5 \mathbb{I} \oplus D_1 \oplus D_4) \\
D_5 &= \text{norm} (k^2 \mathbb{I} C_5 \mathbb{I} \oplus k \mathbb{I} C_1 \mathbb{I} \oplus D_0 \oplus D_2 \oplus D_5 \oplus k \mathbb{I} C_4 \mathbb{I} \oplus D_3 \oplus D_5 \\
&\quad)
\end{aligned}$$

These definitions can now be approximated by replacing any remaining descriptors on the right-hand side (i.e. all D_i) with an evaluation of the nodes' content. This produces level 2 cut off, which is defined as follows:

$$\begin{aligned}
D_0 &= \text{norm} (k^2 \mathbb{I} C_0 \mathbb{I} \oplus k \mathbb{I} C_1 \mathbb{I} \oplus \mathbb{I} C_0 \mathbb{I} \oplus \mathbb{I} C_2 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I}) \\
D_1 &= \text{norm} (k^2 \mathbb{I} C_1 \mathbb{I} \oplus k \mathbb{I} C_0 \mathbb{I} \oplus 3 \mathbb{I} C_1 \mathbb{I} \oplus k \mathbb{I} C_2 \mathbb{I} \oplus \mathbb{I} C_3 \mathbb{I} \\
&\quad \oplus k \mathbb{I} C_5 \mathbb{I} \oplus \mathbb{I} C_4 \mathbb{I}) \\
D_2 &= \text{norm} (k^2 \mathbb{I} C_2 \mathbb{I} \oplus k \mathbb{I} C_1 \mathbb{I} \oplus \mathbb{I} C_0 \mathbb{I} \oplus 2 \mathbb{I} C_2 \mathbb{I} \oplus k \mathbb{I} C_3 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I} \\
&\quad \oplus \mathbb{I} C_4 \mathbb{I}) \\
D_3 &= \text{norm} (k^2 \mathbb{I} C_3 \mathbb{I} \oplus k \mathbb{I} C_2 \mathbb{I} \oplus \mathbb{I} C_1 \mathbb{I} \oplus 2 \mathbb{I} C_3 \mathbb{I} \oplus k \mathbb{I} C_4 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I}) \\
D_4 &= \text{norm} (k^2 \mathbb{I} C_4 \mathbb{I} \oplus k \mathbb{I} C_3 \mathbb{I} \oplus \mathbb{I} C_2 \mathbb{I} \oplus 2 \mathbb{I} C_4 \mathbb{I} \oplus k \mathbb{I} C_5 \mathbb{I} \oplus \mathbb{I} C_1 \mathbb{I}) \\
D_5 &= \text{norm} (k^2 \mathbb{I} C_5 \mathbb{I} \oplus k \mathbb{I} C_1 \mathbb{I} \oplus \mathbb{I} C_0 \mathbb{I} \oplus \mathbb{I} C_2 \mathbb{I} \oplus 2 \mathbb{I} C_5 \mathbb{I} \oplus k \mathbb{I} C_4 \mathbb{I} \\
&\quad \oplus \mathbb{I} C_3 \mathbb{I})
\end{aligned}$$

These definitions can be simplified to produce the following equations:

$$\begin{aligned}
D_0 &= \text{norm} ((k^2+1) \mathbb{I} C_0 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I}) \\
D_1 &= \text{norm} (k \mathbb{I} C_0 \mathbb{I} \oplus k \mathbb{I} C_5 \mathbb{I} \oplus \mathbb{I} C_3 \mathbb{I} \oplus \mathbb{I} C_4 \mathbb{I}) \\
D_2 &= \text{norm} (k \mathbb{I} C_3 \mathbb{I} \oplus \mathbb{I} C_0 \mathbb{I} \oplus \mathbb{I} C_4 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I}) \\
D_3 &= \text{norm} ((k^2+2) \mathbb{I} C_3 \mathbb{I} \oplus k \mathbb{I} C_4 \mathbb{I} \oplus \mathbb{I} C_5 \mathbb{I}) \\
D_4 &= \text{norm} ((k^2+2) \mathbb{I} C_4 \mathbb{I} \oplus k \mathbb{I} C_3 \mathbb{I} \oplus k \mathbb{I} C_5 \mathbb{I}) \\
D_5 &= \text{norm} ((k^2+2) \mathbb{I} C_5 \mathbb{I} \oplus k \mathbb{I} C_4 \mathbb{I} \oplus \mathbb{I} C_0 \mathbb{I} \oplus \mathbb{I} C_3 \mathbb{I})
\end{aligned}$$

In general, level 2 cut off can be defined as:

$$D_i = \text{norm} \left(k^2 \mathbb{I} C_i \mathbb{I} \oplus k \bigoplus_{j \in N} \mathbb{I} C_j \mathbb{I} \oplus \bigoplus_{j \in M} \mathbb{I} C_j \mathbb{I} \right)$$

where

$$N = \{ x \in B \mid \langle x, i \rangle \in L \vee \langle i, x \rangle \in L \} \text{ all immediate neighbours of node } i$$

$$M = \{ x \in B \mid \exists y \in B : (\langle x, y \rangle \in L \wedge \langle y, i \rangle \in L) \vee (\langle i, y \rangle \in L \wedge \langle y, x \rangle \in L) \}$$

*all neighbours of node i with exactly one
intervening node (including return paths)*

Level 2 cut off provides a more complex model of node descriptor calculation. Each node in the document base is now described in terms of its own content, its neighbours' content, and its neighbours' neighbours' content. This produces an algorithm which takes more of the neighbourhood into account and can provide access to a larger set of structures. Non-textual nodes need only be connected by at most two links for a cluster based descriptor to be calculated. With earlier models non-textual nodes had to be directly connected to at least one textual node to provide access by query.

3.9.4 Mixed Descriptor Types

It is possible for document bases to provide a method for indexing particular non-textual media. If this were the case then the cluster based algorithms could be used on all

indexable media. As an example consider the sample network (see figure 3.1) is indexed on textual content and on image impression. Two parallel sets of descriptors can be defined for this network: T_i for textual descriptors and I_i for impression descriptors. These can be defined, using level 1 cut off, as follows:

$$T_i = k \mathbb{I} C_i \mathbb{I}_1 \oplus \bigoplus_{\forall j \in N} \mathbb{I} C_j \mathbb{I}_1$$

$$I_i = k \mathbb{I} C_i \mathbb{I}_2 \oplus \bigoplus_{\forall j \in N} \mathbb{I} C_j \mathbb{I}_2$$

where

$$N = \{ x \in B \mid \langle x, i \rangle \in L \vee \langle i, x \rangle \in L \} \text{ the neighbours of node } i$$

$$D_i = \langle T_i, I_i \rangle \text{ the descriptor tuple}$$

$$\mathbb{I} C_i \mathbb{I}_1 = \text{the indexing of content } C_i \text{ by textual content}$$

$$\mathbb{I} C_i \mathbb{I}_2 = \text{the indexing of content } C_i \text{ by impression}$$

$$\begin{aligned} T_0 &= \text{norm} (k \mathbb{I} C_0 \mathbb{I}_1 \oplus \varphi) \\ &= \text{norm} (k \mathbb{I} C_0 \mathbb{I}_1) \end{aligned}$$

$$\begin{aligned} T_1 &= \text{norm} (k \varphi \oplus \mathbb{I} C_0 \mathbb{I}_1 \oplus \varphi \oplus \mathbb{I} C_5 \mathbb{I}_1) \\ &= \text{norm} (\mathbb{I} C_0 \mathbb{I}_1 \oplus \mathbb{I} C_5 \mathbb{I}_1) \end{aligned}$$

$$\begin{aligned} T_2 &= \text{norm} (k \varphi \oplus \varphi \oplus \mathbb{I} C_3 \mathbb{I}_1) \\ &= \text{norm} (\mathbb{I} C_3 \mathbb{I}_1) \end{aligned}$$

$$\begin{aligned} T_3 &= \text{norm} (k \mathbb{I} C_3 \mathbb{I}_1 \oplus \varphi \oplus \mathbb{I} C_4 \mathbb{I}_1) \\ &= \text{norm} (k \mathbb{I} C_3 \mathbb{I}_1 \oplus \mathbb{I} C_4 \mathbb{I}_1) \end{aligned}$$

$$\begin{aligned} T_4 &= \text{norm} (k \mathbb{I} C_4 \mathbb{I}_1 \oplus \mathbb{I} C_3 \mathbb{I}_1 \oplus \mathbb{I} C_5 \mathbb{I}_1) \\ &= \text{norm} (k \mathbb{I} C_4 \mathbb{I}_1 \oplus \mathbb{I} C_3 \mathbb{I}_1 \oplus \mathbb{I} C_5 \mathbb{I}_1) \end{aligned}$$

$$\begin{aligned} T_5 &= \text{norm} (k \mathbb{I} C_5 \mathbb{I}_1 \oplus \varphi \oplus \mathbb{I} C_4 \mathbb{I}_1) \\ &= \text{norm} (k \mathbb{I} C_5 \mathbb{I}_1 \oplus \mathbb{I} C_4 \mathbb{I}_1) \end{aligned}$$

$$\begin{aligned} I_0 &= \text{norm} (k \varphi \oplus \mathbb{I} C_1 \mathbb{I}_2) \\ &= \text{norm} (k \mathbb{I} C_1 \mathbb{I}_2) \end{aligned}$$

$$\begin{aligned} I_1 &= \text{norm} (k \mathbb{I} C_1 \mathbb{I}_2 \oplus \varphi \oplus \mathbb{I} C_2 \mathbb{I}_2 \oplus \varphi) \\ &= \text{norm} (k \mathbb{I} C_1 \mathbb{I}_2 \oplus \mathbb{I} C_2 \mathbb{I}_2) \end{aligned}$$

$$\begin{aligned} I_2 &= \text{norm} (k \mathbb{I} C_2 \mathbb{I}_2 \oplus \mathbb{I} C_1 \mathbb{I}_2 \oplus \varphi) \\ &= \text{norm} (k \mathbb{I} C_2 \mathbb{I}_2 \oplus \mathbb{I} C_1 \mathbb{I}_2) \end{aligned}$$

$$\begin{aligned} I_3 &= \text{norm} (k \varphi \oplus \mathbb{I} C_2 \mathbb{I}_2 \oplus \varphi) \\ &= \text{norm} (\mathbb{I} C_2 \mathbb{I}_2) \end{aligned}$$

$$\begin{aligned} I_4 &= \text{norm} (k \varphi \oplus \varphi \oplus \varphi) \\ &= \varphi \end{aligned}$$

$$\begin{aligned} I_5 &= \text{norm} (k \varphi \oplus \mathbb{I} C_1 \mathbb{I}_2 \oplus \varphi) \\ &= \text{norm} (\mathbb{I} C_1 \mathbb{I}_2) \end{aligned}$$

This approach can provide many benefits in an environment in which difference media can be indexed in different ways.

4 Justification of Model

4.1 Introduction

To test the hypothesis that cluster-based access to non-textual nodes is a viable option an experiment was carried out using a reasonably large text-only document base; this chapter presents the experiment and its results. The experiment was composed of calculating index-based descriptors (traditional vector-based descriptors) and cluster-based descriptors (based on citation links) for all nodes in the test document base. These descriptors were then compared with each other to establish the level of similarity. To give comparable results the experiment was repeated with randomly created links and levels of similarity were compared with descriptors calculated for citation-based links.

4.2 Test Collection

This experiment was conducted using a collection of 3204 records from the journal, “Communications of the Association for Computing Machinery” (CACM). Each of the records was composed of various fields including title, keywords, abstract, and citations. The citations provide links between all records in the collection which cite records which are also in the collection. The CACM collection was considered suitable for this experiment as it was a reasonable size document base which contained a title for all records and an abstract and keywords for many records. The existence of pre-defined links was required when choosing a test collection, and the citations in the CACM collection provided a set of links which is typical of that found in traditional (non-hypertext) document bases. The CACM collection can be obtained on compact disc as part of a set of collections compiled by E. Fox at the University of Virginia.

The document base contains records of mean size 23.11 terms (standard deviation 21.32). However, the collection is heavily biased towards smaller records with relatively few large records (Figure 4.1 shows the distribution of records by size). The collection also has relatively few records with a large number of links (mean 1.70, standard deviation 3.13) and many records have no links within the text collection at all. To provide suitable comparison material between citation-based links and randomly-created links, the creation of random links was biased towards smaller documents (figure 4.2 shows the distribution of number of links for both citation and randomly created links).

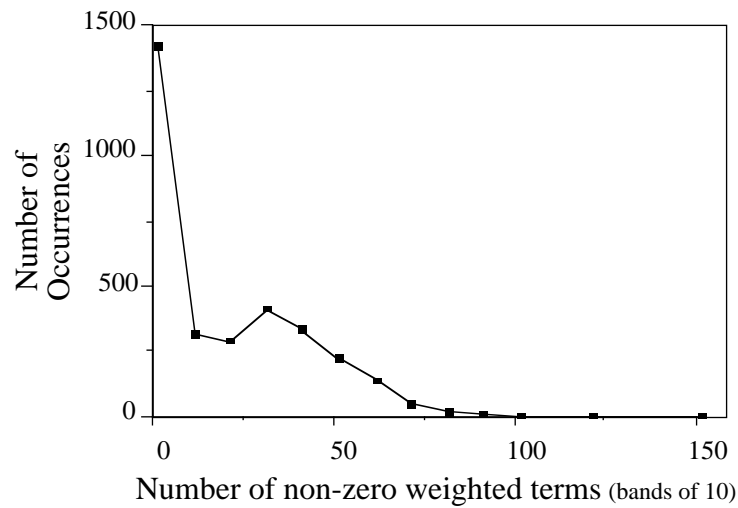


Figure 4.1: Distribution of CACM record size

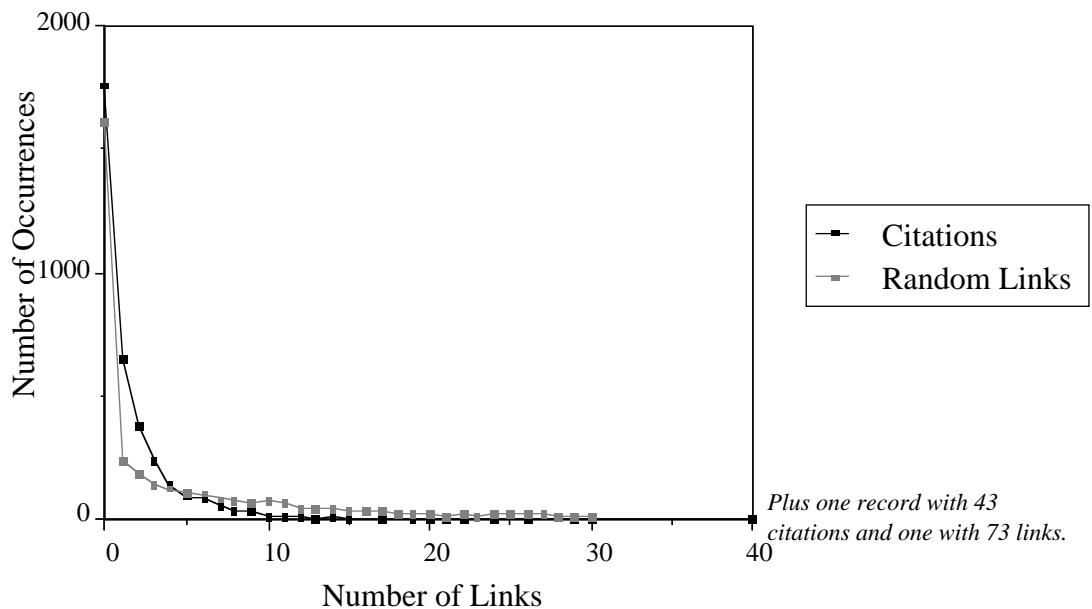


Figure 4.2: Distribution of number of links per CACM record

4.3 Experimental Procedure

This section presents an overview of the process used to test whether cluster-based descriptors of documents do approximate to the meaning of the document. To test this hypothesis a cluster-based descriptor for each record was calculated and compared with the document's index-based descriptor. The results of this experiment were recorded and are discussed in section 4.4.

The entire experiment was conducted on Apple Macintosh computers (a Macintosh IIcx and an SE/30 were used, both are 16 Mhz MC68030-based machines with maths co-processor and fast internal hard disc). The software was entirely written in Think's Lightspeed Pascal by Symantec (Symantec 1988).

4.3.1 Calculating Index Descriptors

To calculate index-based descriptors, the document base was indexed using traditional methods very similar to those used for textual rules in the prototype implementation, *mmIR*, (see chapter 6). Each word in the title, keywords, and abstract fields of the CACM records was initially passed through a stop words filter (van Rijsbergen 1979 pp. 18-19) to remove most of the words which have no meaning when taken out of context (e.g. *and*, *the*, and *not*). The remaining words were conflated to reduce variants of a word to the same stem (Porter 1979). This process resulted in each record being associated with a list of terms and an occurrence count for each term. This information was used to create a list of term-weight pairs in which the weights were calculated using a variation of the inverse document frequency algorithm developed by Sparck Jones and Webster (1980).

$$W_{Di} = \frac{D_i}{D} \log \left(\frac{T}{T_i} \right)$$

where

W_{Di} = weight of term i in record D

T = total number of term occurrences used in the collection

T_i = total number of occurrences of term i in the collection

D = total number of term occurrences in record D

D_i = total number of occurrences of term i in record D

This algorithm takes into account the descriptive power of a term in the document base as a whole and in the current record.

4.3.2 Calculating Cluster Descriptors

A cluster-based descriptor was then calculated for each record based on the simple cluster technique described in section 3.3, this algorithm calculates the average descriptor for the set of records which are linked to or from the current record. This results in a set of descriptors which are based on the context of the record in the hypertext network, the algorithm used can be expressed as follows:

$$D_i = \text{norm} \left(\bigoplus_{j \in N} \llbracket C_j \rrbracket \right)$$

where

$N = \{ x \mid \langle x, i \rangle \in L \vee \langle i, x \rangle \in L \}$ the neighbours of node i

$\llbracket C_j \rrbracket$ = index based descriptor of record j

L = set of pairs representing all links in the document base

$A \oplus B$ = vector addition of vectors A and B

After the creation of cluster-based descriptors most nodes had two descriptors associated with them – one based on indexing (or the document’s content) and one on clustering (or the document’s context). Some records, however, which did not have any citations could not be used to calculate a cluster-based descriptor and consequently could not be used in this experiment – the remaining records which were used in the experiment totalled 1751 or 55% of the test collection.

4.3.3 Comparing Descriptors

Each pair of descriptors (index and cluster based) were compared using the cosine coefficient (van Rijsbergen 1979 pp. 39) to establish how similar they were. The cosine coefficient views the two descriptors as N-dimensional vectors (which start at the origin) and calculates the cosine of the angle between the two vectors. If the descriptors are very similar then the vectors should be close to parallel and have an angle of approximately 0° (cosine=1). Alternatively, two very different documents will have near perpendicular vectors resulting in an angle of approximately 90° (cosine=0). The algorithm can be expressed as follows:

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

where

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^N \mathbf{A}_i \mathbf{B}_i$$

\mathbf{A} = the index based descriptor for record D

\mathbf{B} = the cluster based descriptor for record D

$$\|\mathbf{V}\| = \sqrt{\sum_{i=1}^N \mathbf{V}_i^2}$$

This process resulted in a table of correlations between the two methods of describing a given node. The experiment was repeated using randomly created links instead of citation-based links and an equivalent table of correlations produced.

4.4 Results

After creating lists of correlations between index and cluster-based descriptors for both citation and randomly created links, the results were tabulated and analysed. This showed that the mean correlation was 0.230 (standard deviation 0.182) for citation-based links and 0.037 (std.dev. 0.034) for randomly created links. These results show that citation-based cluster descriptions provide descriptors which are approximately 6 times more similar to the index descriptors than those calculated by the random base case.

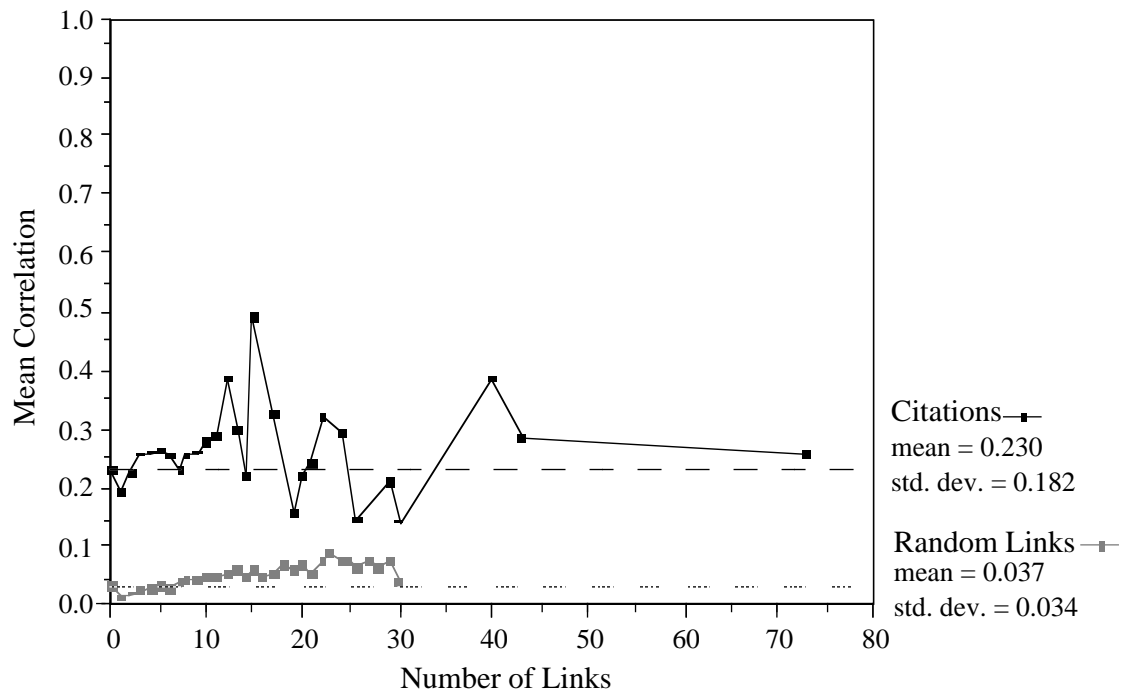


Figure 4.3: Cosine correlation between cluster and index based descriptors (by link count)

Figure 4.3 shows that the mean correlation for citation-based clusters is higher than that for clusters based on random links. It also shows that for the greatest number of records (those with few links) the difference is quite considerable. The mean correlation of random-based clusters appears to rise as the number of links increases. This could be a result of two factors: firstly, the number of occurrences of highly connected records does fall off quite considerably, (see figure 4.2) which will make higher values less accurate. Secondly, it may be an intrinsic feature of the larger descriptors, which will be calculated for large numbers of links. As the size of a descriptor grows (i.e. more non-zero weighted terms are included) the descriptors will start to drift towards the centre of the universe of discourse¹ because the descriptor is starting to describe (the average meaning of) a significant piece of the document base. At the limit the cluster may contain every document in the base, thus describing the average meaning of the entire document base. This should result in the mean correlation being closer than for average document-document comparisons. Figure 4.4 presents an alternative view of the descriptor correlations by showing how they vary in terms of the size of descriptors being compared, again this shows a general rise for random based links as the number of non-zero weighted terms increases.

¹ the document base centroid

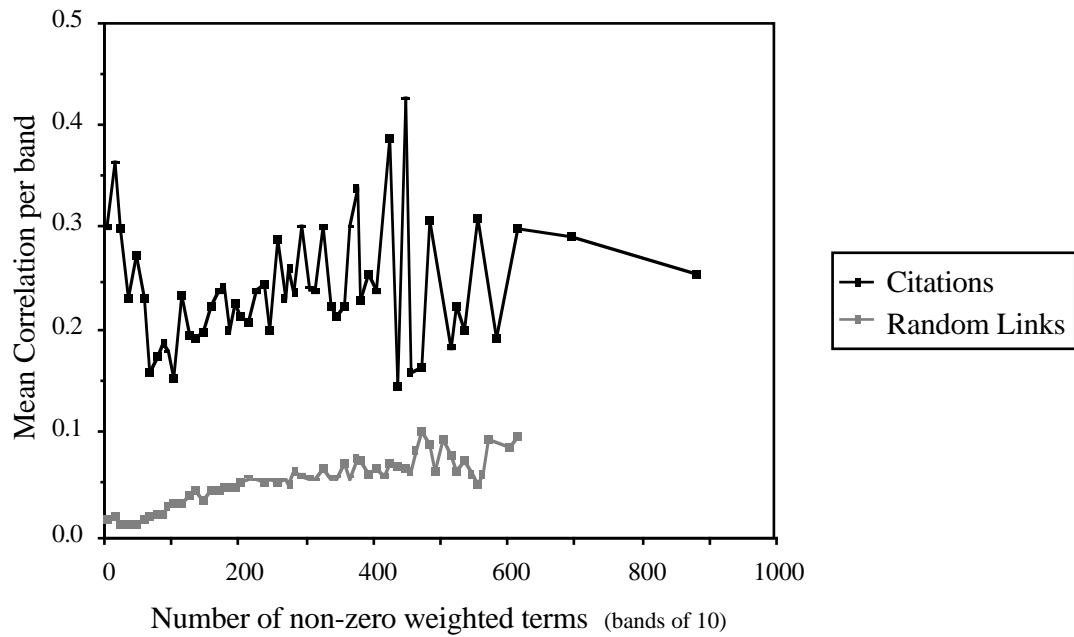


Figure 4.4: Cosine correlation between cluster and index based descriptors (by descriptor size)

4.5 Limitations of Experiment

The experiment carried out within this section has attempted to show that the cluster-based model of descriptor calculation, which was developed in section 3, is worthwhile. To provide a test situation in which the effects could be compared against an automatically derived base case, a text-only document base was used. This raises two issues about the validity of this experiment: will the results extend to a non-textual environment and will links in hypermedia document bases have the same properties as the citations used here?

The usage of the text document base was designed so that when calculating the cluster-based descriptor of a node, its content was completely ignored and the node was treated as if it were non-textual. When considering the calculations carried out with respect to a particular node, the only time its contents were used² were in the calculation of the content-based descriptors for comparison with the cluster-based descriptor. As a result of this, there is no reason why the results shown here cannot be extended to a multi-media environment in which the content-based descriptor cannot be calculated.

A stronger challenge to the validity of this experiment comes when considering the relationship between the citations used here and links in a hypermedia document base. These different forms of connection between *nodes* have two important properties in common: they are created by human users, and there is no single formal definition of the relationship between the two connected nodes. When writing a scientific paper, authors will cite other work for various reasons, for example, citations might be to similar work, contradictory work, interesting work in another field, source for methods used, or for deeper discussions on topics briefly covered. Although much of the work cited by authors

² excluding when it was used to calculate the cluster descriptors for other nodes

is in the same subject area this is not always the case, and citations are not always a strong indicator of the subject content of a paper. Likewise in hypermedia networks, authors will include links to many nodes which they think users might find interesting. The motivation for creating, or following, many of the links in a hypermedia environment would appear to be very similar to that for citations – both are simply connections to something which the reader may find useful or interesting.

It would appear that the relationship between the experimental conditions and those in which the cluster based algorithm will be used are similar enough (in the areas of importance) that the results shown here will be valid in a hypermedia document base.

The relatively low correlations achieved in the experiment will, in part, be due to simplistic retrieval engine which was used in the tests. The retrieval engine was based on simple information retrieval techniques and did not include many features, such as a thesaurus, which would improve the ability of retrieval engine to match documents. These facilities would also improve the correlation between context and content-based descriptors for citation-based clusters. For example, this retrieval engine did not recognise any correspondence between words which would be considered equal if a thesaurus were present. The effects of improving the retrieval engine would have no, or very little, effect of the correlations for randomly created clusters. Thus increasing the difference between random and citation-based correlations.

4.6 Conclusions

This chapter has shown that for the C.A.C.M. collection of 3204 records (of which 1751 were used in the experiment), the cluster descriptors created from citation links were significantly closer to the corresponding index-based descriptor than those created with random links. This provides strong evidence that the cluster-based approach to document descriptors does provides results which are significantly better than random and, therefore, are useful in retrieving documents which cannot be indexed by their own content. Although the correlations for citation-based clusters was still low, this is expected to be higher for more effective retrieval engines. This hypothesis is also supported by observing the behaviour of the prototype application which retrieves images with sensible correlations to given queries.

5 Relevance Feedback

5.1 Introduction

Relevance feedback is potentially the most important part of the user's interaction with a retrieval engine. After users enter their queries they often use relevance feedback to refine their queries so that, eventually, the ideal set of matched documents are retrieved. While chapter 2 gave an overview of relevance feedback, and chapter 7 considers the interface issues involved with feedback, this chapter considers the implications of being able to give feedback on non-relevant documents.

Throughout this chapter the term feedback will be used exclusively to refer to positive feedback, except in section 5.4 which discusses both positive and negative feedback and explicitly states which form is being referred to. Positive feedback is provided by users when they consider that a given node matches their query. Since it permits users to refine their queries without issuing new textual queries, it is a considerably powerful tool.

When users are given access to hypermedia links after issuing a query they are not restricted to viewing relevant documents. It is possible for users to follow trails of links which will take them onto non-matching nodes. This is indeed one of the major benefits of providing users with the query-browsing hybrid discussed in this thesis. As well as being able to view these non-matching nodes, the user is able to give relevance feedback on any node that is encountered. Whereas, in traditional query systems the user is restricted to viewing, and therefore to giving feedback on, nodes which matched the last query¹. An intuitive argument can be made that giving feedback on non-matched nodes should have more effect on the query than giving feedback on matched nodes: if node m matches the latest query then giving feedback on node m could be considered as simply reinforcing the retrieval engine's decision to retrieve that node. Whereas, giving feedback on node m' , which does not match the latest query, could be considered as correcting the retrieval decision. This chapter presents some arguments to back up this intuition and show that feedback on non-matching documents is an important consideration in hybrid systems.

Both justifications given below are based on the vector space model of information retrieval, while section 5.5 briefly discusses the probabilistic model. The approach to relevance feedback used in this chapter takes the vector representing the original query and adds, a fraction of, the vector describing a node which has been stated as relevant. This can be expressed as follows (the symbol \oplus is used here for vector addition).

¹ Some systems do provide access to previous queries thus allowing users to give feedback on non-matching nodes. The results of this chapter apply equally to such systems as to systems built on the hybrid model.

$$\mathbf{q}' = \text{norm}(\mathbf{q} \oplus k\mathbf{d})$$

where

\mathbf{q}' = the new query based on the last query and relevance feedback

\mathbf{q} = the state of the query before the user provides relevance feedback

k = a constant, in the range 0 to 1, defining the strength of feedback

\mathbf{d} = the document which was considered relevant

This model is easily implemented and reasoned about, while being a reasonably good model of relevance feedback (see chapter 2.2.4 for more details). All vectors discussed in this chapter are considered to be normalised, i.e. have unit length. To maintain this invariant the vector representing the new query, \mathbf{q}' , is normalised after vector additions.

5.2 Vector Based Justification

When the user provides feedback on a given document the scaled vector representing that document is added to the vector representing the query. The resulting vector is then normalised to produce a new description of the query. This process can be represented as the addition of two 2-D vectors; although in a retrieval engine the space would be of very high dimensionality, this serves as a useful demonstration. Figure 5.1 shows feedback on two documents, \mathbf{a} and \mathbf{b} , after an initial query \mathbf{q} .

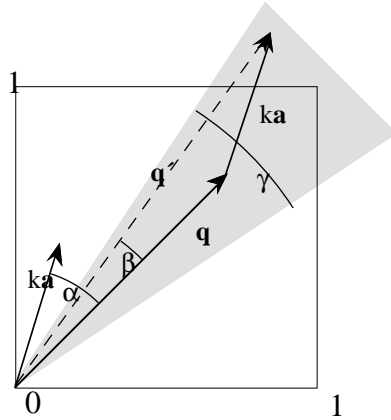


figure 5.1a: Feedback on non-matched node

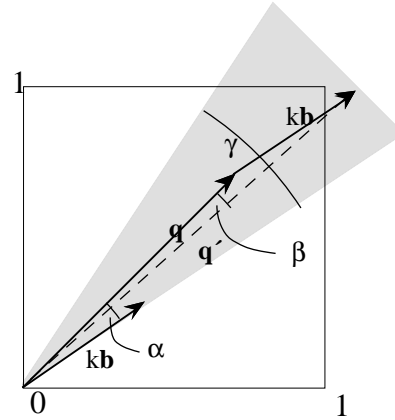


figure 5.1b: Feedback on matched node

figure 5.1: Diagrammatic addition of query vector to feedback vector

In both diagrams the light grey region depicts the area in which any document vectors will have been retrieved in response to the query \mathbf{q} , i.e. if a vector lies within this arc then it is considered as a match to the original query. The area is defined as an angle of γ with the vector \mathbf{q} in the centre. When using the cosine coefficient for retrieval the light grey region is equivalent to the area in which documents are retrieved for a specific cut off, i.e. $\text{cosine_coef}(\mathbf{q}, \mathbf{d}) \geq \cos(\gamma/2)$. The dashed vector, \mathbf{q}' , represents the sum of the query, \mathbf{q} , with the current node vector, \mathbf{a} or \mathbf{b} , before normalisation.

As can be seen from figure 5.1 the angle β is greater, when the angle α is greater. In terms of relevance feedback² this implies that the angular difference between the original and new query vectors is greater when the angular difference between the document and the original query is greater.

The angle β can be calculated as follows:

$$\beta = \tan^{-1} \frac{\mathbf{q}'_y}{\mathbf{q}'_x}$$

where

$$\mathbf{q}' = \mathbf{q} \oplus k\mathbf{d}$$

\mathbf{q} = query vector rotated by σ such that \mathbf{q} lies along the x-axis

\mathbf{d} = relevant document vector rotated by σ

$$\mathbf{q}'_x = \|\mathbf{q}\| + k\mathbf{d}_x$$

$$\mathbf{q}'_y = k\mathbf{d}_y$$

$$\mathbf{d}_x = \|\mathbf{d}\| \cos \alpha$$

$$\mathbf{d}_y = \|\mathbf{d}\| \sin \alpha$$

$$\mathbf{v}_i = \text{the } i \text{ component of vector } \mathbf{v}$$

$$\|\mathbf{v}\| = \sqrt{\mathbf{v}_x^2 + \mathbf{v}_y^2}$$

This can be simplified to define the angle, β , between the query and the query plus feedback as:

$$\beta = \tan^{-1} \left(\frac{\|\mathbf{kd}\| \sin \alpha}{\|\mathbf{q}\| + \|\mathbf{kd}\| \cos \alpha} \right)$$

where \mathbf{q} is the query vector, \mathbf{d} is the document vector upon which feedback is given, α is the angle between \mathbf{q} and \mathbf{d} , and β is the angle between \mathbf{q} and $\mathbf{q} \oplus k\mathbf{d}$. For this definition the vectors need not be rotated to align with an axis, since the definition does not decompose the vectors into constituent parts. The definition is also valid for any dimensionality of space – not just 2D. Since \mathbf{q} and \mathbf{d} are of unit length this can be further simplified to:

$$\beta = \tan^{-1} \left(\frac{k \sin \alpha}{1 + k \cos \alpha} \right)$$

The expression within brackets, above, is guaranteed to be positive since both sine and cosine give positive results when α lies between 0° and 90° . This results in a maximum value for the entire expression when the bracketed expression is maximum. The maximum

² when the lengths of \mathbf{a} , \mathbf{b} , and \mathbf{q} are fixed at respectively k , k , and 1

angle β is thus achievable when $\alpha=90^\circ$. The maximum angle β can then be defined as $\tan^{-1}(k)$, since $\sin(90^\circ)=1$ and $\cos(90^\circ)=0$. This results in a minimum cosine coefficient of $\cos(\tan^{-1}(k))$. The minimum value of β is achievable when $\alpha=0^\circ$, i.e. the query and document are parallel, this yields $\beta=0^\circ$ and a maximum cosine coefficient of $\cos(0^\circ)=1$.

For diagram 5.1 the value of k is taken at approximately 0.5 which is rather larger than normal, resulting in a range of feedback coefficient between 0.89 and 1.0. The effects shown here would also occur with smaller values of k , but the changes would be more subtle and the differences in effect, as well as the effect of feedback itself, would be reduced. The matching angle, γ , is also rather larger than would typically be found – a smaller angle would have no effect on the differences in feedback strength since these differences are only dependent on the pre- and post-query vectors, and not on any cut offs to the list of matched nodes.

5.3 Experimental Justification

To further justify the hypothesis that (positive) relevance feedback has more effect when given on poorer matching nodes a test experiment was run using the CACM collection of abstracts and queries. The experiment proceeded by taking each of the standard queries, giving relevance feedback for each record in turn, and tabulating the effect on the query descriptor. This section describes the process in greater depth and presents the results in section 5.3.3.

5.3.1 Equipment

The test was run on an Apple Macintosh IIfx (a 40Mhz, MC68030 based personal computer) and used the CACM collection of records which is available on a CD-ROM compiled by E. Fox at the University of Virginia. The collection is composed of 3204 records, each record includes title, keywords, and abstract (as well as other fields), the collection also includes a set of 64 standard queries for the set of records. The collection provides a standard environment for assessing the performance of retrieval engines and was also used in experiments to justify the cluster based retrieval of non-textual nodes (see chapter 4).

When a query is matched against each record in the document base there are many more matches with very low coefficient scores than with high scores. Figure 5.2 shows a graph of cosine coefficient scores for matching queries with documents against the number of occurrences of that score. The graph splits the complete range (0..1) of the cosine coefficient into 100 slots and displays the number of matches between queries and records which have a coefficient score within that slot. The lowest value represented by a point is a single occurrence, bands in which zero matches are found are not plotted.

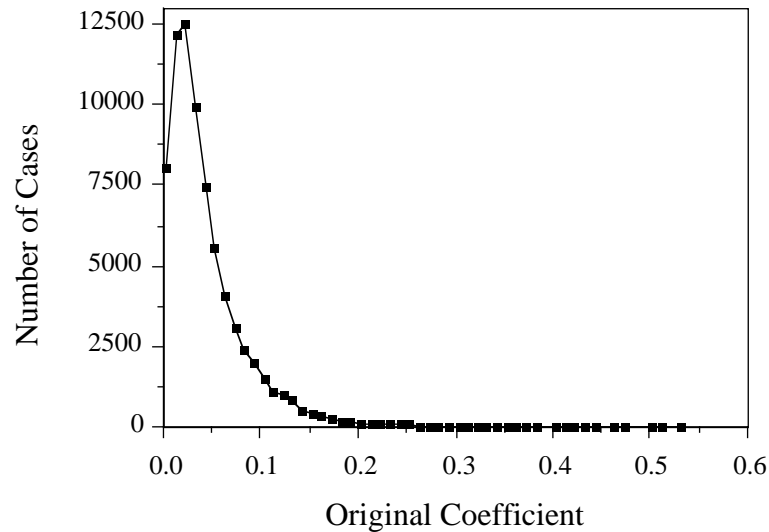


Figure 5.2: Distribution of matching coefficient scores

5.3.2 Process

Initially the CACM collection of records and the collection of standard queries were indexed using standard techniques. The records were indexed by passing them through a standard stop words filter (van Rijsbergen 1979 pp. 18-19). The remaining words were then passed through a conflation algorithm (Porter 1980) to remove inflectional suffixes, and then weighted according to the inverse document frequency algorithm (Sparck Jones and Webster 1980). The queries were processed in a simpler manner by conflating the words and assigning a weight proportional to the number of times each term is used in the query.

To assess the effect of relevance feedback on queries each query, q , was processed in turn and for every record, d , a new query was calculated, q' , as if the user had given positive feedback on that node. This new query was then compared with the original query using the cosine coefficient to establish the similarity of the post-feedback and the pre-feedback queries, and, therefore, how much effect the feedback had on the query descriptor. These results, together with the strength of the original matching, were tabulated and the results are shown in section 5.3.3. The process can be expressed more formally as follows:

$$R = \{ \langle m, f \rangle \mid \forall q \in Q, \forall d \in D: m = c(q, d) \wedge f = c(q, q') \wedge q' = q + kd \}$$

where

R = a set of pairs $(\langle m, f \rangle)$ representing the results of the experiment

m = matching weight between document and query

f = matching weight between post- and pre-feedback queries

Q = set of all query descriptors

D = set of all document descriptors

c = cosine coefficient

k = constant controlling strength of feedback (in experiment $k \approx 1/3$)

5.3.3 Results

The following graph (figure 5.3) shows that the feedback coefficient (the cosine coefficient calculated between the original query and the query plus feedback) does follow a path which increases as the original matching coefficient increases, i.e. the effect of positive feedback decreases as the feedback document becomes more similar to the original query. This supports the intuitive argument which was presented at the start of this chapter.

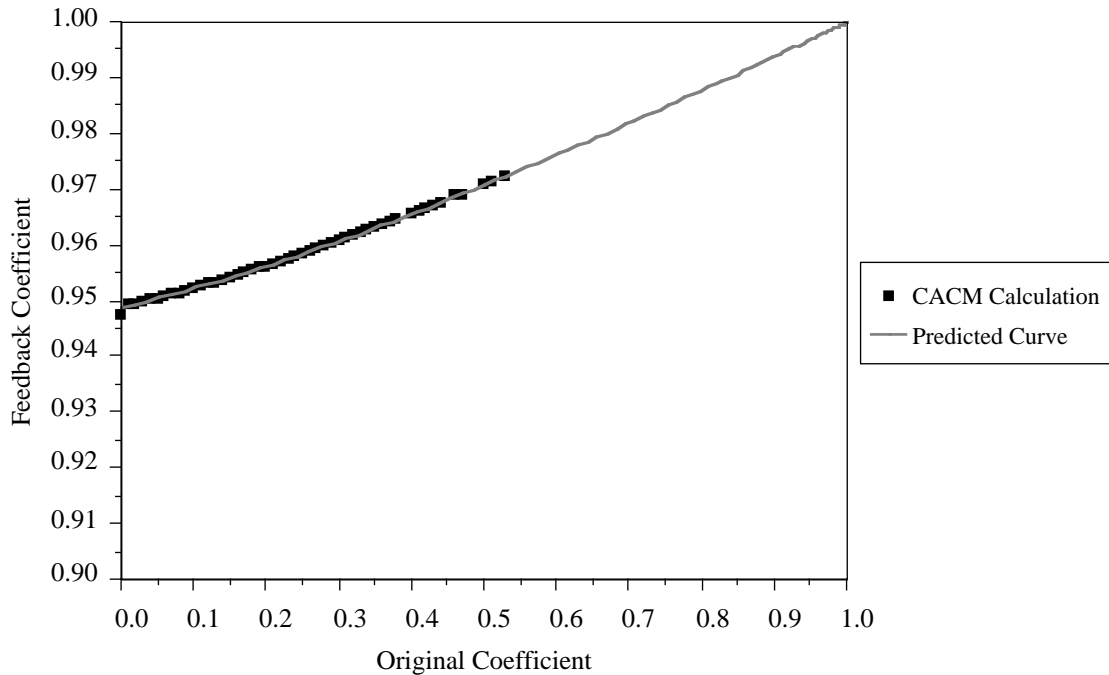


Figure 5.3: Effect of positive feedback on CACM queries

Figure 5.3 shows the results of the CACM experiment as black squares, each point representing the mean value of all records which have the given original matching value (in bands of 0.001, as described in section 5.3.1). The grey line shows the curve produced by calculating the formula $\beta = \tan^{-1}((k \sin \alpha) / (1 + k \cos \alpha))$, as derived in section 5.2, for each integral degree between 0° and 90° – in this graph the values are displayed as $\cos(\alpha)$

against $\cos(\beta)$. In both cases k was taken as 0.333. Although the range of values is quite small, 0.949 to 1.000 and approximately 0.95 to 0.97 over the used range of original matchings, the range of feedback effect will be more pronounced when using a higher value of k – see figure 5.4.

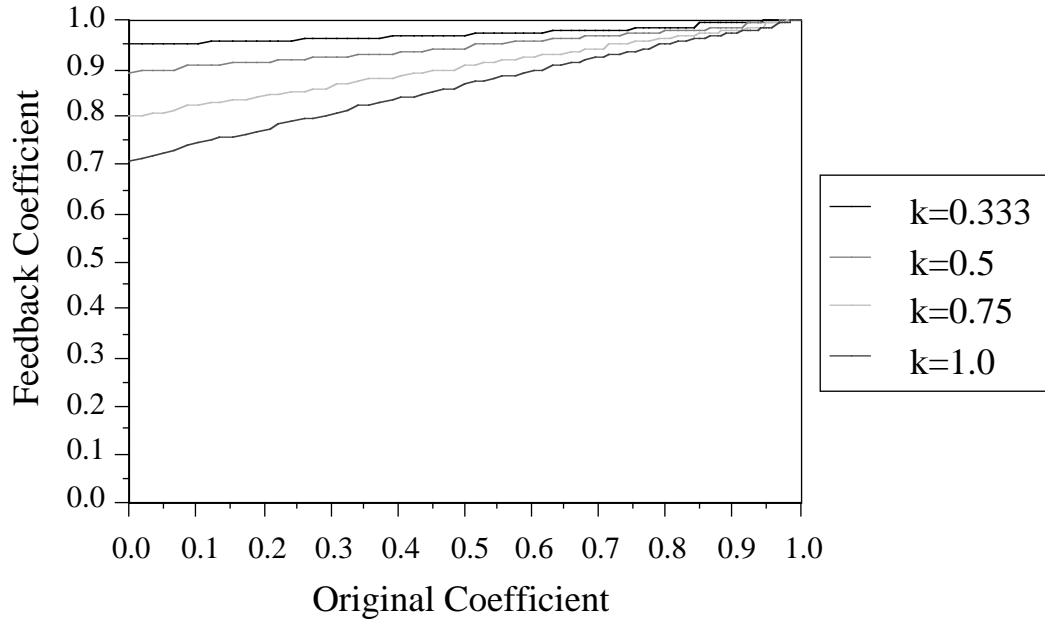


Figure 5.4: Effect of positive feedback for various values of k

5.4 Negative Feedback

Although chapter 7 shows that users can have a very poor understanding of negative feedback, some systems provide it and the effects of giving negative feedback must be considered. Users should provide negative feedback when they consider the current node to be totally irrelevant. This information can then be used to provide better separation of relevant and non-relevant documents when the user next issues a query command. Following the intuitive argument set out at the start of this chapter for positive feedback, it would be reasonable to expect that giving negative feedback on non-matched documents should be considered as less important than giving negative feedback on matched nodes. Negative feedback on a non-matching node could be expected to have very little effect since the feedback is only reinforcing the retrieval decision, while negative feedback on matched nodes should have a larger effect since this is an attempt to correct errors by the retrieval engine. Unfortunately this intuition is not correct when considering the vector-based model of information retrieval. Figure 5.5 shows the effect of giving negative feedback on two nodes (\mathbf{a} which does not match the current query and \mathbf{b} which does). Negative feedback is modelled by stating that $\mathbf{q}' = \mathbf{q} - k\mathbf{d}$, for these diagrams $k \approx 0.5$.

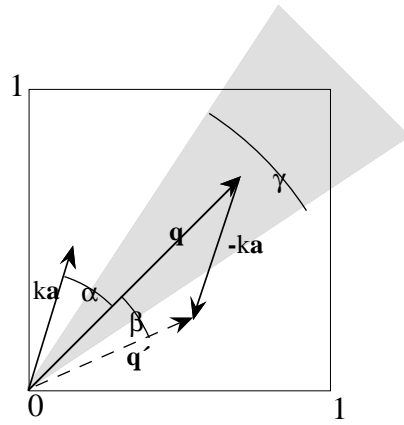


figure 5.5a: Feedback on non-matched node

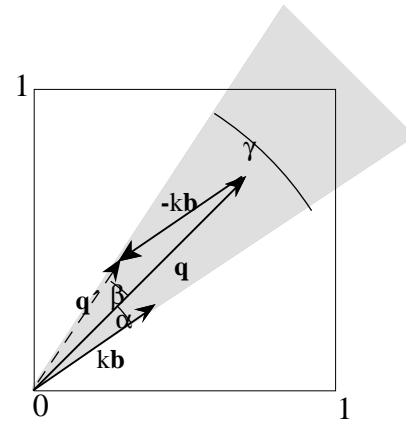


figure 5.5b: Feedback on matched node

figure 5.5: Diagrammatic subtraction of query vector from feedback vector

As can be seen from figure 5.5, providing negative feedback on a non-matched node, *a*, has a much greater effect on the query than providing negative feedback on a matched node, *b*. The CACM collection experiment was also run using negative feedback. The results from this test were plotted in figure 5.6 (black squares) together with the curve which is predicted from the equation $\beta = \tan^{-1}((k \sin \alpha)/(1 + k \cos \alpha))$ where $k = -0.333$.

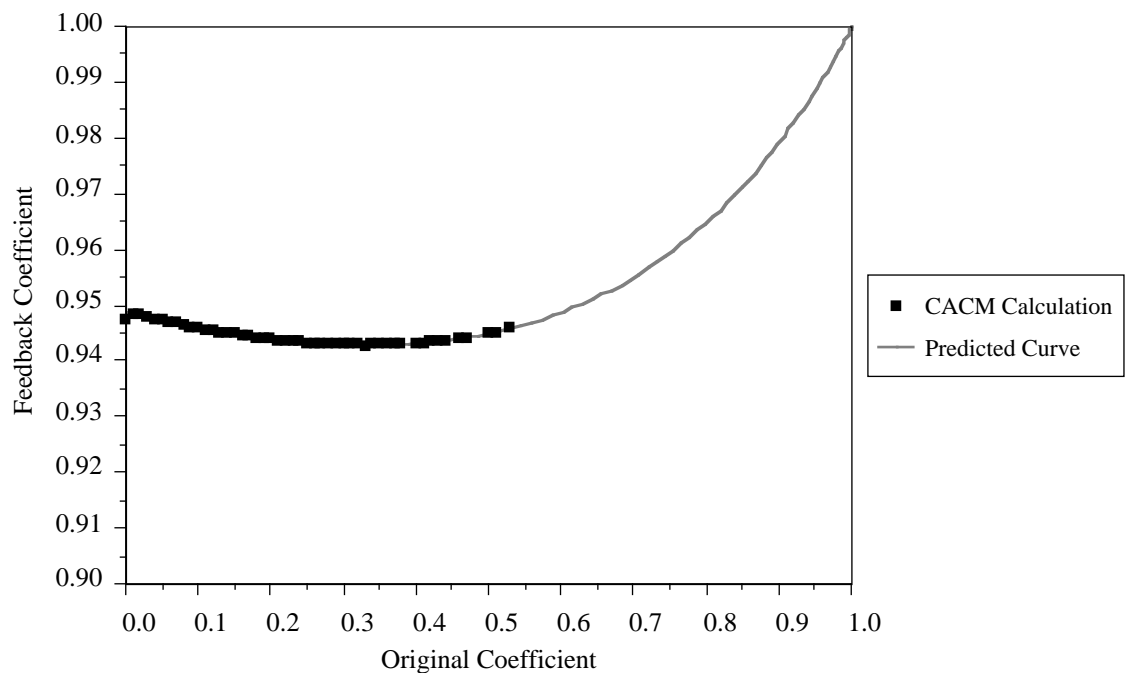


Figure 5.6: Effect of negative feedback on CACM queries

As can be seen from the graph the similarity between the original query and the query plus negative feedback increases, in general, as the original matching score increases. This occurs to the extent that when the original query is a perfect match, the user giving negative feedback has no effect. This does not support the intuitive argument discussed at the beginning of this section, and indeed shows a significant imbalance in the treatment of positive and negative feedback. The effects, however, may not be apparent in a real situation since the effect of feedback is almost constant in the range 0.0 to 0.6 which is the

range in which the majority of document – query matches fall. The turning point in this range is due to the angle β starting to reduce as α approaches 90° . Figure 5.7 shows the effect for larger values of k .

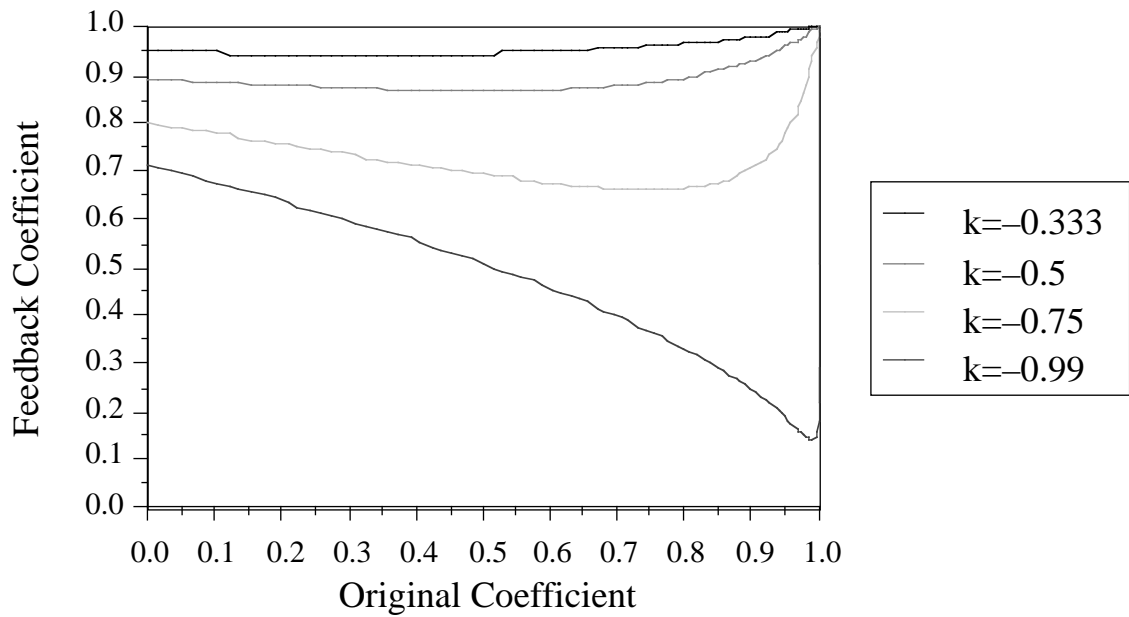


Figure 5.7: Effect of negative feedback for various values of k

Figure 5.7 shows that the effect of negative feedback is far from the inverse of positive feedback. Firstly, the slope of the curve is initially downwards, this is expected since the closer the original match the greater the effect that would be expected for negative feedback. However, the graphs contain a turning point after which closer original matches result in a smaller effect on the query. This turning point does, however, lie towards the top range of original matching coefficient and such strong matching coefficients are unlikely to occur regularly within a document base. With this in mind a more significant problem may be the overall strength of negative feedback and the effect of negative feedback on very poorly matching original documents. The overall effect of negative feedback is much stronger for values of k more negative than -0.5 than for the equivalent positive feedback. As an example consider $k=0.75$, the largest effect of positive feedback results in a feedback coefficient of approximately 0.80 whereas a feedback coefficient of approximately 0.65 can be achieved with $k=-0.75$. Although the minimum value for the feedback coefficient lies within the high range of original matching the overall effect of negative feedback is stronger than for positive feedback. Another major consideration of negative feedback is the effect on the query when users give feedback on a completely irrelevant document. Although it can be argued that an effect should occur on the query for this form of feedback, no equivalent effect occurs with positive feedback on perfectly matching documents. It is this imbalance between negative and positive feedback that may lead to the largest problems in feedback usage: if a user gives negative feedback on a node,

issues a query and then gives positive feedback on the same node the query will not be restored to the original state.

5.5 Probabilistic Model

The discussion presented in this section has considered the fixed increment error correction method of providing relevance feedback in a vector-based retrieval system. The effects produced for positive feedback are as expected, with the possible exception of there being no effect on perfectly matching nodes. The initial intuition may reasonably be assumed to translate to the probabilistic model of retrieval. When the user marks a document as relevant, which the retrieval system had categorised as irrelevant, this will change the underlying base of known relevant documents. It will have a more significant effect on future queries than if the user added a document which was predicted as relevant. The effects for negative feedback, which were shown here, would not occur within a probabilistic model of retrieval: a user adding a document to the list of non-relevant documents, which was predicted to be relevant, will have considerable effect on the estimations of probability which will be made for the next query. Whereas, stating that a predicted non-relevant document is non-relevant will have little effect on the estimations of probability. It is also expected that a probabilistic model will provide a relationship between negative and positive feedback which is significantly closer to an inverse relationship.

5.6 Conclusions

This chapter has shown that the effect which a user giving feedback on a document has on the query does depend on how well the document matched the original query. The tests have shown that the effect of positive feedback diminishes as the matching between the document and the original query increases. This matches the intuitive argument which can be made by considering positive feedback as providing reinforcement of the retrieval decision, when the retrieval engine correctly retrieves a relevant document, or providing correction, when it failed to retrieve a relevant node. It is expected that this effect will translate from the vector space model, which was used for these tests, to other models of information retrieval.

Unfortunately negative feedback was shown, under the vector space model, not to coincide with the intuitive argument. Over the regularly used range of matching values (say 0..0.7), the effect of negative feedback does increase as the matching between the original query and the document increases. However, as the original matching approaches a perfect match the effect starts to decrease until there is no effect for negative feedback on a perfect matching document. Intuitively one would expect that the effect of negative feedback would increase continuously as the original matching increased, since at low matching values the user is supporting the retrieval decision. At high matching values the user is attempting to correct the decision and would expect a significant change in the

query representation to achieve this. However, in the lower range of matching values (0..0.7) in which the vast majority of document – query matches occur, the effect of negative feedback is decreasing, so this imbalance between positive and negative feedback may not be significant in practice. A more significant problem, which will occur in practice when large feedback factors are used, results from the imbalance in strength between negative and positive feedback. Negative feedback typically has a much larger effect on the query than positive feedback. This may be avoided by using a smaller feedback factor for negative feedback than for positive feedback. However, negative feedback and positive feedback cannot be considered as inverse operations under the vector space model because of the size of the effects each produce and the shape of the effect curves. The effects shown in this chapter will also occur query-only systems which permit relevance feedback on non-matched documents, for example, some systems allow users to give relevance feedback using documents from previous queries. It is not expected that the results for negative feedback will translate to sounder models of information retrieval, which should give effects for negative feedback which are closer to initial expectations.

6 mmIR – A Prototype Implementation

6.1 Introduction

A prototype implementation, known as the MultiMedia Information Retriever (*mmIR*), was developed to test the usefulness of the hybrid model of information retrieval developed in chapter 3 and to assess the user interface. This chapter describes the implementation and how access can be gained to the document base through traditional free text retrieval, hypermedia browsing, and a hybrid access based on the two models. The chapter also briefly describes the internal design of the application.

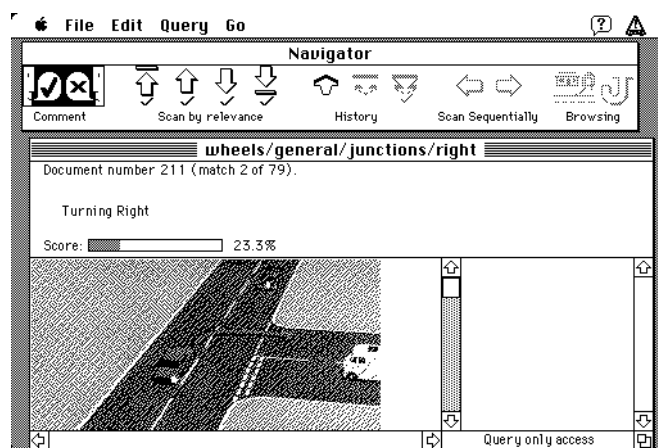


Figure 6.1: A typical mmIR screen

6.2 Document Base

mmIR uses the British Highway Code (Department of Transport 1988) as a test document base. The Code is composed of 198 short rules, each of which describes a particular aspect of driving safely and legally within the United Kingdom. Associated with some of these rules are images which give extra details or clarify the textual descriptions: these images are required in conjunction with the text to achieve a full understanding of the Highway Code. The entire set of rules was indexed together with 21 associated images. Query-based access to images was provided through the basic clustering technique described in section 3.3.

The Highway Code was chosen as a test document base for several reasons: firstly, the Code is composed of many small textual nodes with a reasonable number of associated images, which makes it almost ideal for testing a free text retrieval engine and for testing the cluster description technique. The Code also contains many links in its paper form (e.g. rule 169 describes motorway driving in fog by stating that “it is vital that you should obey the rules in Rule 55 [Driving In Fog]”). These links provided a basic hypermedia

network which could be used for browsing access, each node has a mean of 0.358 (std. dev. 1.43) links to other rules/images in the Highway Code. Figure 6.2 shows the distribution of links throughout the Highway Code, although the majority of nodes do not have links and many only have one or two, the document was considered to contain enough links to provide access by browsing (when supplemented with nearest neighbour links). When nearest neighbour links were added 128 rules were assigned an extra link based on its similarity with another document in the document base (see section 6.4.5). These links are not included in the following graph or in the average given above.

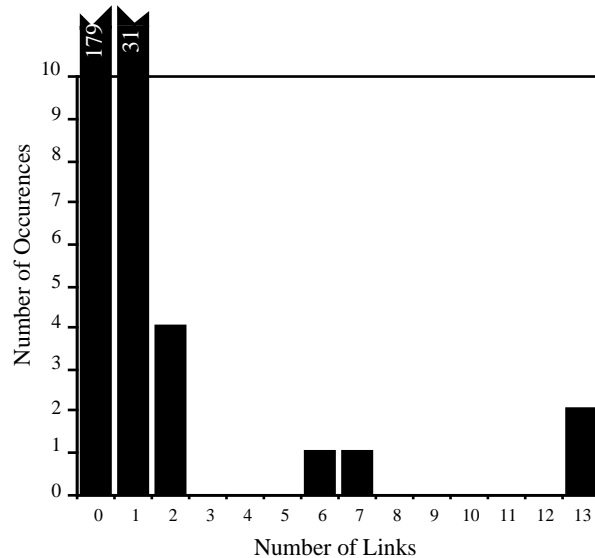


Figure 6.2: Distribution of Links

The size of nodes in the Highway Code was also considered suitable for testing; the rules were neither too short to index accurately nor too long to read quickly. The rules had an average of 327.6 characters (std. dev. 313.6), with approximately 5.6 characters per word this gives an approximate average of 59 words per rule. Figure 6.3 shows the distribution of node size, in terms of characters, the graph shows bands of 100 characters (e.g. the bars represent 0...99 characters, 100...199 characters etc.).

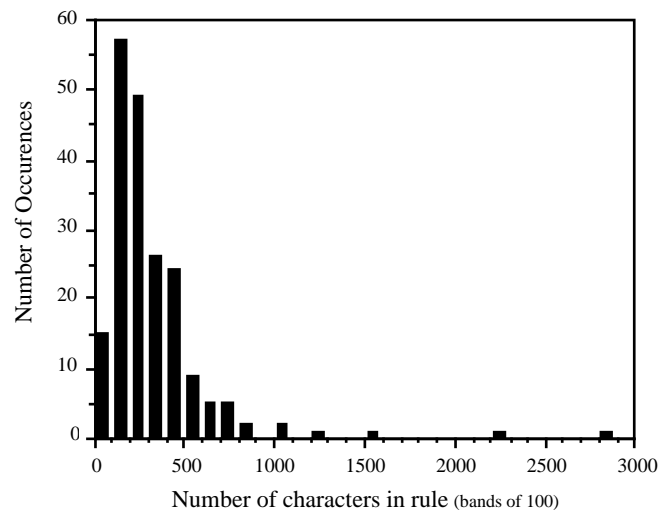


Figure 6.3: Distribution of node size in the Highway Code

The Highway Code is also well known to a large percentage of the population and so was considered suitable for user testing (see chapter 7).

6.3 Access

Access to the Highway Code was provided via three variants of the interface:

- Query-only access
- Browsing-only access
- Combination of query and browsing access

The three access methods are all variants of the same interface – various parts of the display are hidden and various commands are not available in each mode. The following three subsections describe each variant of the user interface.

6.3.1 Query-Only Access

To access the Highway Code using query-only access users must initially enter a query. When the application starts up a window is displayed asking for a free text query to be entered (figure 6.4). After entering this query there is a delay while the retrieval engine matches the query's descriptor against the descriptors of rules and images in the Highway Code.

Please enter your query as free text, all documents considered relevant to your query will be retrieved when you hit OK.

Figure 6.4: Query window

After the query has been processed a list of matched nodes is created and the top of this list (the best match) is displayed in the node window (figure 6.5). This window shows the content of the current node in the bottom left-hand corner, it also shows information about that node in the top section of the window. The bottom right section is used for displaying links and is therefore not used in this variation of the interface.

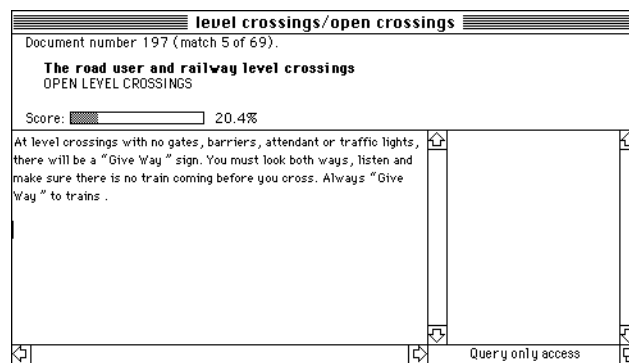


Figure 6.5: Node window for query access

At the top of the information section is a display of the document (or node) number for the currently displayed node. To the right of this are details of how far down the list of matched nodes the current node is. The matched node list is ranked, so users need only scan as far down the list as required to satisfy their search. In the case of figure 6.5 the user is not expected to scan 69 nodes. It is envisaged that users will only scan at most the top ten best matches before deciding to re-formulate the query, either directly or through a relevance feedback based query.

Below the node number is the node's title. The title can be composed of up to three lines representing the major section, section, and sub-section of the Highway Code; the window's title bar presents a summary of the node's title.

Finally, at the bottom of the information section is a display of how well this node matches the last query, this score is given in the form of a bar graph and in textual form as a percentage of the best possible match. Some systems present a score based on a relative scale in which the best matching document scores 100%. Here the score is calculated on an absolute scale between 0% and 100% where 100% is achieved by comparing a document against itself and scores are not relative to the best match.

Navigating The Nodes

The application has a navigator window (figure 6.6) which is used to control access to the list of matched nodes together with other regularly used commands.



Figure 6.6: Navigator window for query access

The “scan by relevance” group of four vertical arrows, towards the left of the navigator window, allow users to move to the best, the next better, the next poorer, and the worst matched node (respectively from left to right). This is the only method available for users to peruse the matched node list.

To the right of the “scan by relevance” controls are the history mechanism controls. These allow the user to backtrack through nodes that (s)he has visited and then undo the effect of backtracking by coming forward through the history list. The rightmost history command is not used during query-only access. The history information is erased when the user issues a new query, so (s)he is restricted to looking at nodes viewed since the last query; the mechanism also has a fixed maximum number of entries (during the user tests this was set at 50, with the list pruned to 40 when the 51st entry was attempted). These restrictions are enforced for technical reasons, however, they should not affect the users of the small test document base.

The two comment icons at the left of the navigator window allow users to indicate, using relevance feedback, that the current node is relevant (happy man) or is non-relevant (sad man) to the current query. After users have browsed around the matched node list giving relevance feedback they can issue a relevance information based query by choosing the “query using relevance information” menu option (see figure 6.7), this issues a new query based on a combination of the original query plus any relevance information which the user has provided.



Figure 6.7: Query menus

At any point during the information search the user can issue a new textual query by choosing the “Query By Text” menu command. This brings up the query window (figure 6.4) with the previous query displayed, and selected, so that the user can alter it or enter a completely new query.

The scan sequentially, browsing, and rightmost history commands are not available to users of query only access and are shown greyed-out.

6.3.2 Browsing-Only Access

The browsing-only interface initially displays the *home* node, this node contains the most general sections of the Highway Code and is shown in figure 6.8. This is an example of a header node which is used to provide access to the Highway Code by browsing.

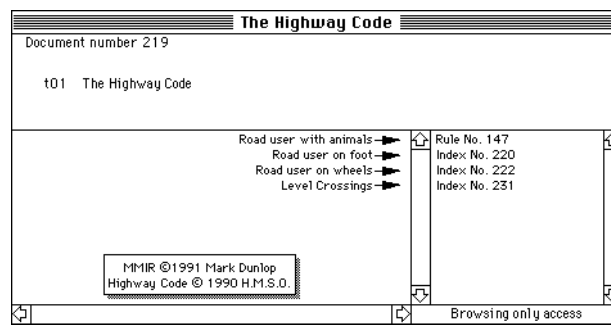


Figure 6.8: Node window for home node

The information section and the node display section of the window are approximately the same as described in section 6.3.1 for query-based access. The bottom right section of the window is, however, used for displaying the links which can be followed from the current node (by double clicking the appropriate line). Header nodes provide information about the destination of links within the node display section of the window, however, rule or image nodes use this area for their content and hence provide no information about the link except that which is contained within the node itself (see figure 6.9).

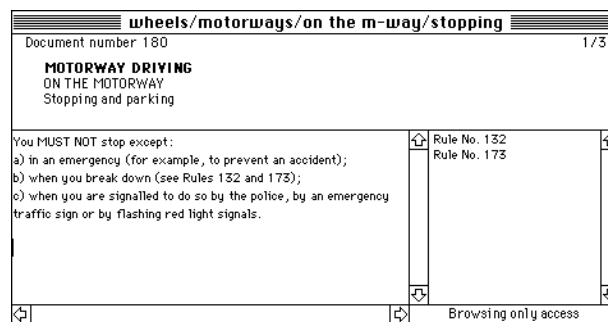


Figure 6.9: Node window for browsing access

The information section of the window contains the same details as for query-only access but without references to the matched node list (since there is no such list). The only additional item is the page number at the top right of the window. This states the current page number and the total number of pages within the current sub-section of the Highway Code: e.g. 1/3 states that this is page one of three pages in this sub-section.



Navigating The Nodes





Figure 6.10: Navigator window for browsing access

The main form of navigation when using browsing-based access is by following links. There are several forms of link used, the most common is the reference link which represents the links that existed in the paper-based Highway Code. These are shown in the link list as either ("Rule No. X" or "Picture No. X"). In addition to the links in the

paper-based Highway Code there are various links to provide access to pictures. In the paper-based version most pictures are implicitly referenced by their juxtaposition with the appropriate textual rules. These links are all uni-directional, that is, there is no automatic method for following a link of this type backwards, to the source from the destination. This type of link was chosen because they are similar to the links used in the paper based Highway Code; uni-directional links are also more common in hypermedia systems than bi-directional links and so are more representative.

Structural links are also provided for all the nodes in a particular sub-section of the Highway Code. In the paper-based version links to the previous and next nodes are provided implicitly by the ordering of the rules. Within the hypermedia implementation these links are implemented in the form of next and previous arrows ( ) which, unlike the paper based variation, restrict the user to moving back and forth within the current sub-section. This limitation provides a more natural hypermedia structure by removing the single path through the entire document base.

The index header nodes (figure 6.8) contain a list of links to nodes (“Rule No. X”) or to other header nodes (“Index No. X”). These links could be considered as bi-directional since their effects can be undone by using the *go to parent* command () which takes the user from the currently viewed node to the index node that provides access to it. A variation of this command is the *go home* command () which takes the user directly to the home node.

The final type of link is created automatically by the retrieval system. *Similar to* links provide access to the retrieval engine without the requirement for a query. Each node in the Highway Code has a nearest neighbour calculated for it. This operation calculates which node in the document base has a descriptor which is most similar to the descriptor of the current node. If the match between the node and nearest neighbour is close enough then a similar to link is added to the links for that node. The algorithm which was used to calculate these links is described in more detail in section 6.4.5. The provision of nearest neighbour links could be extended to define, for each node, a list of near neighbours. The advantages of this extension were considered small compared to the increase in complexity for users, hence *mmIR* created, at most, a single nearest neighbour link.

The history mechanism is identical to that described for query-based access, except that history details are only erased when the size of the go-back path exceeds the maximum size of 50 nodes.

The scan by relevance commands and the rightmost history command are not available to users of browsing access since there is no list of matched documents. To reflect this the commands are shown greyed-out.

6.3.3 Hybrid Access

The final variation of the interface provides hybrid access through a combination of hypermedia browsing and free text querying. Together with the ability to query the

document base, hybrid access provides users access to links within the Highway Code (references, forward and backwards links). However, users of the hybrid model are not given access to *similar to* links nor to the hypermedia header nodes. These restrictions were imposed to simulate, more closely, the facilities which would be provided in a true hybrid retrieval system. Some of the major benefits of this approach are concerned with authoring the document base, consequently a full set of header nodes is unlikely to exist within a hybrid system. Since the user has direct access to the retrieval engine, it is also unlikely that similar-to links would be provided within a hybrid system.

When the application starts in this mode users are presented with a query window (see figure 6.4) into which a free text query is entered. This results in a list of matched nodes (identical to that for query-only access) and the best match is displayed. A node window contains all of the information shown on the node windows for browsing and for query-based access: title, node number, matching position, matching score, the current page number, and the size of the current sub-section (an example is shown in figure 6.11).

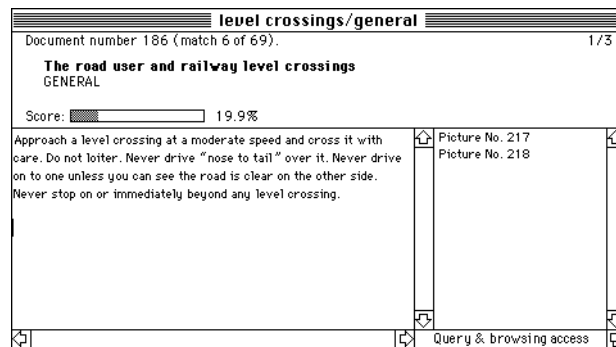


Figure 6.11: Node window for hybrid access

The navigator (figure 6.12) also provides all the functions available in both browsing and query-based access with the exception of the *go parent* and *go home* commands which are not available and are shown greyed-out.



Figure 6.12: Navigator window for hybrid access

The hybrid model presents the user with a rather complex model of the document base. The primary concern of users is the matched node list which is returned from their queries. While scanning down this list they may, however, be side tracked by following links onto nodes which did not match the last query. It may also be possible for users to follow links off the matched node list and then to re-enter the list at a different location (see figure 6.13).

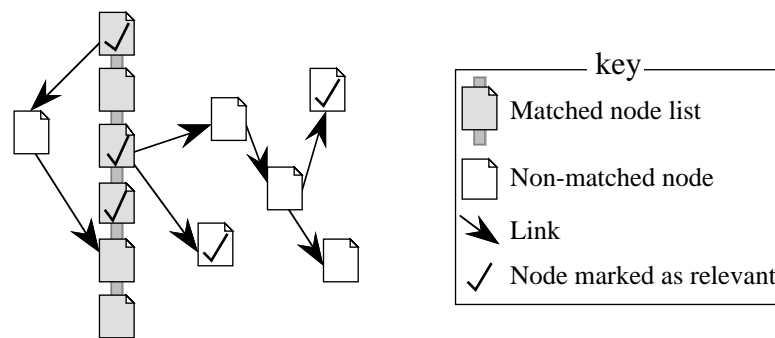


Figure 6.13: Browsing in the hybrid model

While users are navigating through the hybrid document base they can provide relevance feedback on any visited nodes, whether it matches or does not match the query. This differs from traditional relevance feedback systems which only permit the user to view (and hence give feedback on) nodes which matched their latest query.

To help users with orientation problems after leaving the matched node list an addition history command (🏠) is available when users are viewing a non-matched node. This command issues consecutive go back commands until the currently viewed node matches the latest query (this command is not available if there has not been a query or if there were no matches to the query). The matching score and matched node position are not displayed when viewing a node which does not match the latest query.

6.4 Internal Design

This section describes, briefly, algorithms and methods used by *mmIR*, the information should be detailed enough to permit a system being developed using the same model and methods.

6.4.1 Indexing Textual Nodes

The rules in the Highway Code were indexed into two files: ‘Word Occurrences’ and ‘Node Descriptors’. The word occurrences file lists each term together with a list of nodes it occurs in and a count of how often it occurs in those nodes. Whereas, the node descriptions file stores the converse information: one record per node stating the terms which occur in that node and how often they occur.

Before terms were entered into the index files they were passed through a filter to remove stop words; these are words which have no meaning when taken out of context and, thus, cannot be used by a term based retrieval engine to differentiate between relevant and non-relevant nodes. The stop words list which was used was taken from van Rijsbergen (1979 pp.18-19).

The words which were not removed by the stop words filter were then conflated using Porter’s algorithm (1980). This algorithm attempts to reduce all forms of a word to the same stem: for example *connect*, *connected*, *connecting*, *connection*, and *connections* all conflate to the term *connect*.

The file structures can be expressed in extended Backus-Naur form (EBNF) as follows¹:

```

WordOccurrences2 ::= { WordRecord } END_OF_FILE
WordRecord3 ::= { Occurrence } END_OF_RECORD
Occurrence ::= NodeNumber OccurrenceCount

NodeDescriptions4 ::= { RuleRecord } END_OF_FILE
RuleRecord ::= RuleNumber WordUses
WordUses5 ::= { WordUseRecord } END_OF_RECORD
WordUseRecord ::= Word OccurrenceCount

```

Where Word is a string no longer than 15 characters, NodeNumber and OccurrenceCount are numbers in the range 1...198, END_OF_RECORD has value -1, and END_OF_FILE represents the logical end of the file data structure.

6.4.2 Image Indexing Algorithm

The images were indexed by taking a vector sum of the descriptors of the nodes which are connected to them, and then normalising the vector so that it has a length of 1. This is the same process as described in section 3.3, and can be defined as:

$$D_i = \text{norm} \left(\bigoplus_{j \in N} C_j \right)$$

where N is the set of textual nodes which neighbour C_i

This produces descriptors which are dependant on the content of nodes which are immediately adjacent, in the hypermedia network, to the picture. The average number of rules which are connected to an image node is 1.35 (standard distribution 0.75), with the majority of image nodes having a single link pointing to them. This resulted in the majority of images being indexed in a similar manner to traditional free text retrieval systems which provide access to images. However, the indexing was considerably faster since the text did not need to be written, but the descriptor is likely to be a less specific description of the image since the text was not hand produced. The remainder of the 21 picture nodes (four which were pointed to by two links, and two which were pointed to by

¹ The small subset of EBNF used here can be defined as follows: $A ::= B C$ states that A is defined as B followed by C . $\{A\}$ states that A may occur zero or more times.

² the list of word records which compose a word occurrences file are ordered alphabetically on Words

³ the list of occurrences which compose a word record are ordered on NodeNumbers

⁴ the list of rule records which compose a node description file are ordered by number. There are 198 rule records composing a node description file – one for each rule in the Highway Code.

⁵ the list of word use records which compose a word uses are ordered alphabetically on words.

three links) did have sufficient links for the cluster-based approach to work effectively, by averaging the content of the nodes which point at the image.

6.4.3 Query Processing

Query Indexing

The query is indexed by a very similar routine to that used for creating the index files. The string is initially converted to lower case and any non-alphabetic characters are dropped⁶. Each word taken from the resulting query string is passed through the stop word filter and then through the conflation routine. Each term (or conflated word which has been used to describe a node) is added to the query descriptor with a weight proportional to the number of occurrences of the term, and its stem equivalents, in the query.

Matching Algorithm

After the query has been indexed into the same format as the nodes in the network, each node's descriptor is compared with the query's descriptor. A variation of the matching algorithm developed by Croft & Harper (1979) is used to perform this comparison.

The matching process can be summarised as follows:

$$L = \text{Sort} (\{ \langle w, ID \rangle \mid ID \in D \wedge w = m(D_{ID}, D_Q) \})$$

where

w = *weight of the matched document*

ID = *identifier of matched document*

L = *matched document list*

D = *all IDs used in the document base*

D_Q = *descriptor of the current query*

Sort = *produces an ordered list in decreasing order of weight*

m = *the matching algorithm*

The matching algorithm can be defined as follows:

⁶ This removes all punctuation but, unfortunately, also removes digits (such as 30MPH). This would not strongly effect the retrieval performance but should be corrected for future versions of the software.

$$m(\mathbf{A}, \mathbf{B}) = \frac{C P_1 + P_2}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|}$$

where

$$P_1 = \sum_{x \in N} \sum_{y \in N} \frac{A_x}{\langle \mathbf{A} \rangle} \frac{B_y}{\langle \mathbf{B} \rangle}$$

$$P_2 = \sum_{x \in N} \sum_{y \in N} A_x B_y \log \left(\frac{(W - W_a)}{W_a} \right)$$

$$\|\mathbf{V}\| = \sqrt{\sum_{i \in N} V_i^2} \quad \text{length of the vector } \mathbf{V}$$

$$\langle \mathbf{V} \rangle = \sum_{i \in N} V_i \quad \text{number of words in document } \mathbf{V}$$

W = number of words⁷ in the document base

W_i = number of occurrences of word i in the document base⁸

C = constant defining importance of each part of formula for mmIR $c=0.75$

6.4.4 Relevance Feedback

When a user provides relevance feedback on a given node, all non-zero weighted terms in that node have the corresponding weight in the query descriptor increased by a constant value. This method is known as fixed increment error correction (see section 2.2.4 and van Rijsbergen 1979 pp. 107) and provides an effective and very fast method of implementing user feedback. The relevance information is not taken into account immediately but only when the user next issues a relevance based query. This approach allows a user to process a list of matched nodes by scanning through it and giving feedback on some nodes. After this scan, if the user is still unsatisfied, the relevance information can be used to provide a new (hopefully improved) list of matched nodes.

The relevance feedback algorithm can be expressed as follows:

$$\forall i \in N : q'_i = k_i$$

where

$$\begin{array}{ll} k_i = q_i & \text{if } d_i = \emptyset \\ q_i + c & \text{if } d_i \neq \emptyset \text{ and feedback is positive} \\ q_i - c & \text{if } q_i \neq \emptyset \text{ and feedback is negative} \end{array}$$

where q'_i is the value of the i th dimension of query q after feedback is given, N is the total number of terms in the document base, i.e. the dimensionality, d_i is the document on which feedback is given, and \emptyset is the empty descriptor, i.e. a descriptor which cannot match any query.

⁷ Here *words* is taken to mean occurrences of words, in their various forms, which are indexed. The total number of words is the count of the number of occurrences of any form (via Porter's algorithm) of a word which is used in the indexes.

⁸ in its various forms as calculated using Porter's algorithm.

This algorithm adds the terms which are used in the node, which was marked, to the query with a fixed weight. If the term already exists in the query then its weight is increased or decreased as appropriate. Since inverse document frequency is used when comparing the query with each node's descriptor, the weights added to the query did not have to reflect the strength of each term to discriminate between documents. It might, however, be favourable if the weights added to the query reflected how descriptive that term was of the marked node. This could be performed by defining k_i in terms of how often term i occurs in d .

6.4.5 Nearest Neighbour Calculations

To calculate the *similar to* links, which are used in the browsing-only interface, the nearest neighbour for each node in the network was calculated. This process involves calculating for each node which node provides the highest matching value with it. The set of nearest neighbours can be expressed as:

$$NN = \{ \langle i, j, m \rangle \in D \times D \times R \mid \\ j \in (D - i) \wedge \nexists k \in (D - i - j) . \text{matching}(k, i) > m \wedge m = \text{matching}(j, i) \}$$

Where D is the set of node IDs used in the document base and R is the set of real numbers. The matching algorithm used in the implementation was the same as that used for matching queries to nodes in the network (see 6.4.3). The set of nearest neighbours was then filtered to remove any distant neighbours, that is, nodes whose nearest neighbour matching was below a given level, this results in the set of nearest neighbours which were added to the document base, NN' , which can be defined as follows:

$$NN' = \{ \langle i, j, m \rangle \in NN \mid m \geq t \}$$

Where t is a threshold value. In the implementation the threshold was set at 0.01 (or 1%). The low figure of 1% was chosen to give access to nodes which, though not on a very similar subject, would potentially be of interest to the user. This filtering results in a set of links which should provide useful cross references in the hypermedia environment. 128 links were created of this type giving 59% of the nodes in the document base a nearest neighbour link. The user interface was developed so that nearest neighbour-based links were not shown if another link exists between the two nodes.

Nearest neighbour-based links were only calculated for textual rules, since images within the Highway Code are mostly calculated by use of a single link from a textual node. These images would be guaranteed to have the node from which they were calculated as their nearest neighbour, providing only a *go-back* link. It is also likely that, in general, many of the image nodes with multiple sources would have their nearest neighbour within the set of nodes used to calculate their descriptor. As a result of

considering this, the process of calculating descriptors for non-textual nodes was considered too similar to the calculation of nearest neighbour links to be of use. With hindsight however, the inclusion of nearest neighbour links would be useful for non-textual nodes whose descriptors are not calculated from a single node. Even if the nearest neighbour was within the set of nodes used to calculate the node's descriptor, it would provide an alternative path when the user enters the non-textual node from one of the other source nodes. The techniques of cluster-based non-textual access and nearest neighbour links, could be beneficially used together in an environment where most cluster-based descriptors are calculated from more than one linked node.

6.4.6 History Mechanism

The major problems facing users of large hypermedia networks are often navigational. Hypermedia relies upon users browsing through the network in search of nodes that satisfy their information requirement. However, when users 'gets lost' there are usually few facilities to aid users recover to a position where they can carry on searching. One of the most common, and basic, navigational aids is a history mechanism. This provides users with a facility to role back the browsing sequence until they recognise their location and can start searching again.

A history mechanism was implemented for *mmIR* which is based on two stacks: a back-to and forward-to stack. When users are navigating (without use of the history mechanism), each visited node is added to the back-to stack. When users decide to go back to a previously visited node a three-step operation is performed: firstly the current node is added to the forward-to stack, then the top of the back-to stack is popped and becomes the current node. The back-to stack builds up until a new textual query is issued; it is also restricted to a fixed number of entries (currently the stack is pruned to 40 nodes whenever the 51st entry is attempted). The double stack approach permits users to have (theoretically) infinite retrace facility and the ability to go back to the start of the information search and scroll forward to replay the choices made. An example use of the history mechanism is given in figure 6.14.

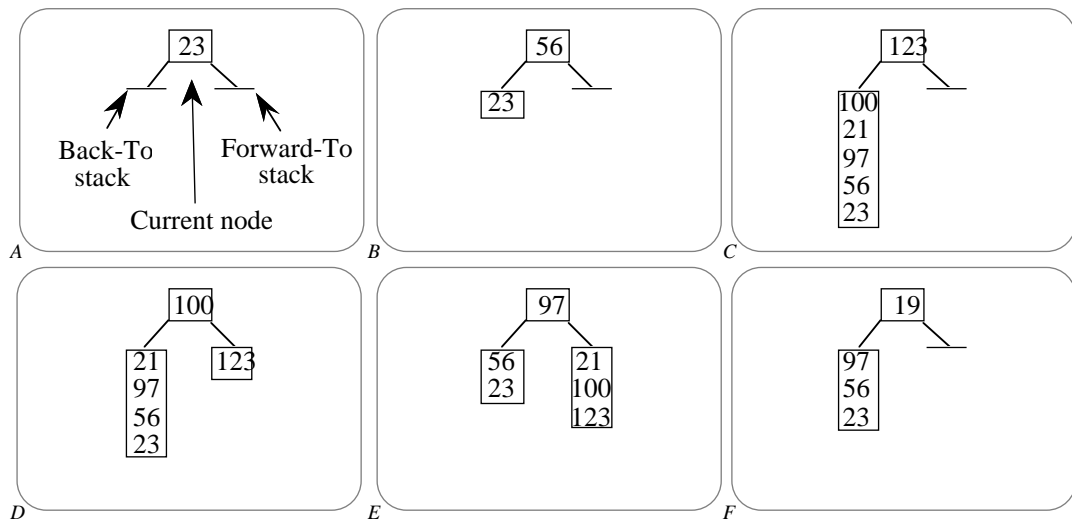


Figure 6.14: Example use of double stack history

In figure 6.14A, a user has just started an information search and is looking at node number 23, at this stage both the go-back and the go-forward stacks are empty. The user then moves from node 23 to node 56 resulting in node 23 being added to the go-back stack (figure 6.14B). A further four nodes (97, 21, 100, and 123) are visited before she decides to use the go-back command and retrace her steps (figure 6.14C shows the status of the history mechanism just before the go-back command is used). When she issues the go-back command the current node is placed on the go-forward stack, the go-back stack is popped, and the popped node is made the current node (figure 6.14D). The user then retraces her steps until she reaches node 97 (figure 6.14E) when she changes path and visits node 19. The diversion from the history path results in the previous node being added to the go-back stack, the current node being set to the newly visited node, and finally the emptying of the go-forward stack (figure 6.14F). The go-forward command is not covered by this example, however, its actions are identical to the go-back command except the direction of pushing and popping is reversed.

Although it was envisaged that the history mechanism would be of most use to users of the browsing-only interface it was made available to all users (see chapter 7 for details of test users usage of the history mechanism). Within the hybrid user interface an extra history command was provided to go back to the last matched node, this provides users with a method for returning to the matched node list after a digression into neighbouring nodes. The command is implemented as a series of consecutive go-back commands until the current node has a non-zero score.

6.4.7 Data Acquisition

The Highway Code was entered through a flat bed scanner (produced by Apple Computer) equipped with various pieces of software. The textual rules were scanned by character recognition software (Caere's OmniPage) and then spell checked to remove the few errors in the character recognition software's output. Images were scanned using

Apple's AppleScan software and were stored in monochrome bitmap format for use on monochrome computers and in four bit greyscale format for use on greyscale or colour computers. The greyscale images were then edited using a colour painting program to artificially add some colour to the images. Apart from the manual addition of some colour, the greyscale images were not edited. The monochrome images did, however, require touching up: in particular any text within the monochrome images had to be added by hand since the original bitmap text was unreadable.

6.4.8 Software Development

mmIR was written entirely on Apple Macintosh personal computers. The development took place on a Macintosh Plus, IICx, and an SE/30 – the Plus was an 8 MHz 68000 based computer, while the IICx and SE/30 were both 16MHz 68030 based computers (with floating point co-processors). Think's Lightspeed Pascal from Symantec (version 2.0) was used as the sole programming language, none of the code was written in other languages or imported from application creators. Lightspeed Pascal provides a superset of Macintosh Pascal, which is itself a superset of standard Pascal. As well as providing full access to the Macintosh toolbox (built-in routines for controlling the Macintosh user interface and hardware), it provided support for object-oriented programming – much of the code for *mmIR* was written in this variation of object pascal. Although far from a perfect object-oriented language it did provide many of the benefits of object oriented programming. This version of pascal also provided extensive support for strings (a major failing of standard Pascal) and a very easily, and extensively, used unit facility which included individual compilation. The final code which was used for user testing accounted for 27 units of Pascal and approximately 10 500 lines of code.

The Macintosh operating system stores many of the user interface specifications within resource files so that they may be altered at a later date without requiring re-compilation. For example, all strings which are shown to the user should be kept in the resource file so that the application can be translated to another language without the need for changes to the code and re-compilation. Menus, windows (document and dialogue, e.g. the navigator), strings, and controls (e.g. scroll bars) were all defined within resources by use of Apple Computer's ResEdit (version 1.2 and later version 2.1).

6.5 Suggested Improvements

The current implementation presents the results of a query as a hidden list of matched nodes which the user can only navigate via the up/down/top/bottom commands. The interface would be considerably improved if this list were made more explicit so that users can more easily associate with the list. One very successful method of making the matched node list more explicit is to present it through a header window, this window would only contain brief details of each node and a link to each node. This method was used to great effect by Sanderson (1990) in a retrieval system for newspaper archives. It was felt that

the provision of this style of header would bias the user testing since it would provide a greater level of navigational aid for query-based access than is provided for hypermedia access, which only had basic navigational aids. However, this would have provided a greater integration of the two access methods. This argument is discussed in greater depth along with other interface issues in chapter 7.

The representation of the score could be improved to give users more information about the results of their query. It is not clear whether the score of the best matched document should be 100%, and all other matches are scaled accordingly, or whether all matching values should just be given directly. This issue is also discussed more in chapter 7.

The general level of navigational aids is rather low in *mmIR*, some form of data overview would provide useful help to users navigating the environment. In the hypermedia arena common navigational aids include navigational maps (see section 2.3.6 for more details), however, the implementation of these techniques would not be trivial if the resulting software was to run on a standard 21cm Macintosh screen.

The cluster technique used for calculating the descriptions of non-textual objects is based on the simplest model described in chapter 3. It is expected that improved descriptions would be created for most nodes by use of level 1 cut off model of the clustering techniques. Considerably improved descriptions should be created for image nodes if the level 2 cut off model were used as this takes more distant nodes into account and would thus reduce the number of nodes with identical descriptions to the connected textual node.

One of the major failings of the retrieval system is a lack of a thesaurus. For the system to be used on a wider basis than the prototype a comprehensive thesaurus should be provided. A simple thesaurus would table all the words used to index nodes and a list of synonyms for each word. When users issue a query which contains a word which is not used for indexing but is a synonym of one which is known, the word would be replaced by then known word. More complex thesaurus systems assign weights to the synonyms to provide more flexibility (for example *disk* and *disc* are identical and would have an synonym weight of 1, whereas *truck* and *lorry* would have a lower weight, and *bus* and *vehicle* a still lower weight).

In the current implementation all relevance information is disposed of when users issue a new text-based query. It would be useful if the option to retain this information was given so that terms can be added to the textual description as required without starting a completely new query. Relevance information is not displayed when a user clicks on a comment icon, the provision of a visual statement within the node to state that it has been marked as relevant would provide an improved understanding of relevance feedback. This issue is discussed further in chapter 7.

6.6 Conclusions

This chapter has presented an overview of *mmIR* together with a deeper description of some parts of the implementation. The prototype has shown that it is possible to construct an information retrieval system based on the hybrid model and for this model to be accessed through different interfaces. It has also shown that this process may be carried out on a reasonably low-powered computer: *mmIR* can run on a one megabyte Macintosh Plus, and runs at a perfectly acceptable speed on a Macintosh SE/30.

7 User Testing

7.1 Introduction

The prototype application, *mmIR*, was tested on a group of 30 users to attempt to establish how users use the retrieval system, for example, how fast they are at reaching their goals, and how confident they are that the goals have been satisfied. The three variants of access to a hybrid document base (browsing, querying, and a hybrid of querying and browsing) were tested and their results compared. These tests also provided some evidence for which directions should be followed when developing a user interface to a document base which incorporates techniques from free text retrieval and from hypermedia, and some suggested improvements are made at the end of the chapter.

7.2 Experiment Overview

Each of the users spent approximately one and a half hours performing a total of 14 small tasks. The tests were split into two sessions, the first session was composed of eight questions and took approximately one hour, the second session, which was scheduled one week after the first, was composed of six questions and took approximately thirty minutes. Before the first session users were asked to fill out a brief questionnaire to judge their experience of the Highway Code, computers in general, and of the Macintosh. At the end of both session the users were asked to complete a questionnaire which assessed how confident they felt about their answers and how long they felt it had taken to answer the questions. At the end of the second session users were also asked to complete a questionnaire which attempted to gauge how well they understood the workings of the system.

During the test itself, users were encouraged to talk aloud to express what they were doing and what problems they were having – very little input was made to the session by the interviewer. The users actions were logged automatically by the software and extra comments were taken on paper by the interviewer. All the subjects used exactly the same software on the same computer. The operating system was, however, upgraded during the trial period but this did not affect the user interface or performance times so should have had no effect on user performance.

The subjects were mainly students and had used a Macintosh for word processing (and perhaps other activities), but they had either no, or very little, experience of programming or of computing science. Adverts were placed in various public buildings on campus (e.g. libraries and unions) to attract students. The subjects were given a £10 book token to entice enough students to take part.

7.3 Questionnaires

The tests were mainly conducted by use of questionnaires – these provided a fast method of testing the users ability, and provided a good level of consistency between the different access methods. In each session the user was given a one page guide to using the software (which was specialised for the access method), together with a four page questionnaire - the first page of which simply gave instructions and space for the user's name and address. The remaining pages are discussed in the rest of this section, examples of these questionnaires are given in section 7.11 (at the end of this chapter). The results of these questionnaires are presented in section 7.6.

7.3.1 Prior Experience Questionnaire

The first questionnaire which users had to fill out was presented at the first session and attempted to judge each user's level of experience. To assess the users experience with the test document base, each user was asked to assess their knowledge of the Highway Code (on a scale between 'none' and 'very good') and how long they have held their full licence (or if they have not passed, how long they have been taking lessons). To assess their level of competence with computers the users were asked to state how often they used a computer (on a scale from 'never' to 'most of my working day') and how well they knew the Apple Macintosh.

Although this questionnaire was short it provided all the information which was required to assess how well users could be expected to perform on the first couple of tasks (i.e. their novice performance). Further questions may have been useful to assess whether users had used any information retrieval or hypermedia software, but these may have influenced the way users approached a system and therefore questions of this nature were not included.

7.3.2 Test Questionnaire

The actual tests consisted of a list of questions (eight for the first session and six for the second) which were in increasing order of difficulty (approximately). Users were simply asked to 'find the most appropriate rule in the Highway Code' for each question and write the rule's document number as the answer to the question. The requirement for users to write the document number down as the answer forced them to use the retrieval software, rather than simply answer the questions from their own knowledge. The first three questions of the first session were of similar difficulty and so could be used to judge the initial learning curve of users. The last question of the first session was repeated as the last question of the second session. Although not a thorough test, this did provided a direct method for comparing how the users had improved between the mid-point of the test and the end. For each access method three questionnaires were created so that any unforeseen difficulties with specific questions / questionnaires would not have a great effect on the

overall performance for that access method. Overall, the questions increased in difficulty over the both sessions with the start, and end, of each session being of approximately the same difficulty. The questions were selected from a set of 14 randomly chosen questions from a book of 300 questions (Humphries 1987), an extra fifteenth question was excluded from the tests because the answer lay within a non-indexed appendix of the Highway Code.

During the tests users were not prohibited from taking notes. However, notes were only taken on a couple of occasions by users noting a possible answer before continuing their search.

7.3.3 Performance Questionnaire

At the end of each session the users were asked to fill out a questionnaire on how they and the computer performed during the test. Users were initially asked to state how many of their questions were not answered, and then to separate the remaining questions into six categories of confidence (on a scale from ‘definitely correct’ to ‘definitely wrong’). If the users failed to answer any questions they were asked to state how confident they felt that there was no relevant information in the Highway Code. These two questions were designed to assess how confident the user was in her/his performance. One of the traditional problems of information retrieval systems is that the user is never quite sure if they have found all the information they require. Although this is a different question to that addressed in these tests, since users are asked to find only one document, it was hoped that these questions would give an impression of how confident users felt. The results of these questions can then be compared with how well the user actually performed in the tests.

The users were then asked how many steps they took to answer the questions (on a seven point scale between ‘much fewer than expected’ and ‘many more than expected’). The meaning of *step* was deliberately not specified since this question was attempting to gauge how fast the user thought (s)he had reached the answers, the answers could be compared with the actual number of nodes visited or with the time taken. To attempt to assess how the speed of the computer inhibited the activities of the users, they were then asked how they found the computer’s response time (answers on a scale between ‘much slower than expected’ to ‘much faster than expected’). To provide a feel for how impressed the users were with the overall performance of the software they were asked whether they could do better with the computer software or with the paper based booklet. Although not a substitute for performing paper-based test, this question was hoped to give some insight into how users rated their overall performance.

After the test questionnaires had been marked, users were asked to state how surprised they were by their performance – this provides an indication of how accurate the users categorisation of answers (into correctness bands) was.

7.3.4 Understanding Check

At the end of the second session users were asked to fill out a quick questionnaire which would assess their knowledge of the system. The questions on this last form were based mainly on topics which are of more general interest than the comparison between the three access methods, for example users of query based access methods were asked to describe how they thought the retrieval system matched queries against documents. This questionnaire was conducted in a less formal manner, especially the browsing access version which was discussed rather than being treated as a form.

7.4 Sessions

The sessions were all conducted in a small office with the user, the interviewer taking notes, and on some occasions a couple of additional people using the office who were not connected with the tests. Users were encouraged to talk aloud while performing the tests, and were encouraged to give comments when they were having difficulties. The interviewer was very careful to give no information which may help the users or bias their use of the system. The only advice that was widely given was in use of the query window: many users were not aware of the standard Macintosh techniques for manipulating text in such a window, especially on their second and subsequent searches when the previous query was displayed for editing or deletion. The users were simply told that they should remove the previous text by pressing the delete key, or that the text could be edited in the same fashion as MacWrite¹. Many users were also informed that they should start a new query by selecting the “By text” entry in the “Query” menu. At the end of the first session there were three users who had failed to grasp a major piece of their access method. They were given guidance on their failing at the end of this session. There were two browsing-only users who had not learned to use the horizontal arrows (◀ ▶) to move through a section of the code, and one user of query-only access who had not learned to use the vertical arrows (⏮ ⏭) to navigate the matched document list was informed of their usage. These failures were considered so significant that they should be corrected before the start of the second session so that all users could be assumed to be reasonably competent when starting the second session, no users of the hybrid access method had failed to learn at least one of the main access methods.

7.4.1 User Experience Results

The users were initially asked to assess their knowledge of the Highway Code, the results showed that on average users estimated their knowledge to be reasonable. The prior knowledge of the Highway Code did vary slightly between the users who took the three different access methods. Figure 7.1 shows the mean values (points) together with the standard deviation (horizontal lines centred around the mean value).

¹ a very popular Macintosh word processors, especially for novice users

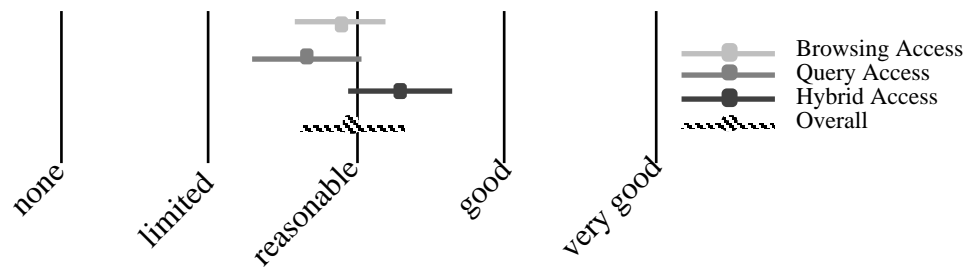


Figure 7.1: Graph of prior knowledge of the Highway Code

These results show there is a slight difference in knowledge for those users who used each access method. The slight differences in prior knowledge did not appear to be a significant factor in the users' performance. This question also shows that overall users considered they had reasonable understanding of the Highway Code. This was supported by a mean number of years driving of 3.85 (with a very high standard deviation of 5.98, caused by some very experienced drivers), the majority of users (62%) had driven for less than two years (where driving time before passing test was counted at half the rate than after passing, e.g. a user who has not passed the test but had been learning for 2 years was considered to have 1 years driving experience). The remaining users had mostly passed within ten years, there were also two users with 12 years driving, one mature student with 30 years experience, and two users who had no experience of driving who claimed they had limited (but not zero) knowledge of the code.

Experience of computer usage again showed slight variations amongst the users who took part in each access method, but it is unlikely that these differences would be strong enough to affect the results. Figure 7.2 shows the results, again with mean values displayed as points and standard deviations as lines centred around the mean.

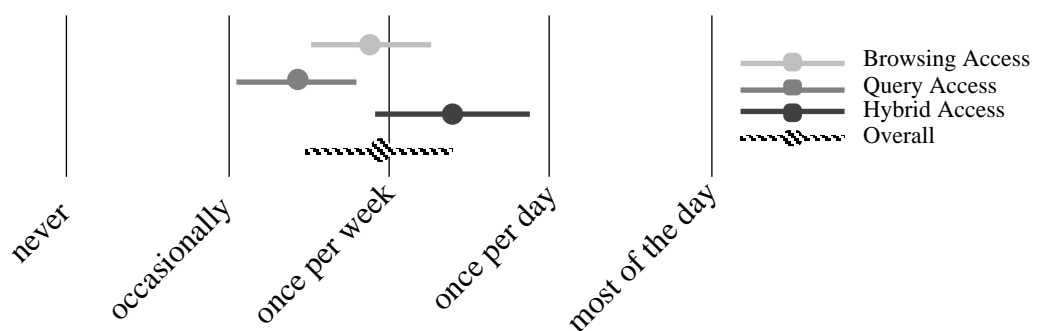


Figure 7.2: Graph of computer usage

This showed that the average user used a computer once per week, no users had never used a computer, 72% of users used the computer either occasionally or once per week, the remainder used the computer once per day with one user using it for most of the working day. Figure 3 shows how well users stated that they knew the Apple Macintosh.

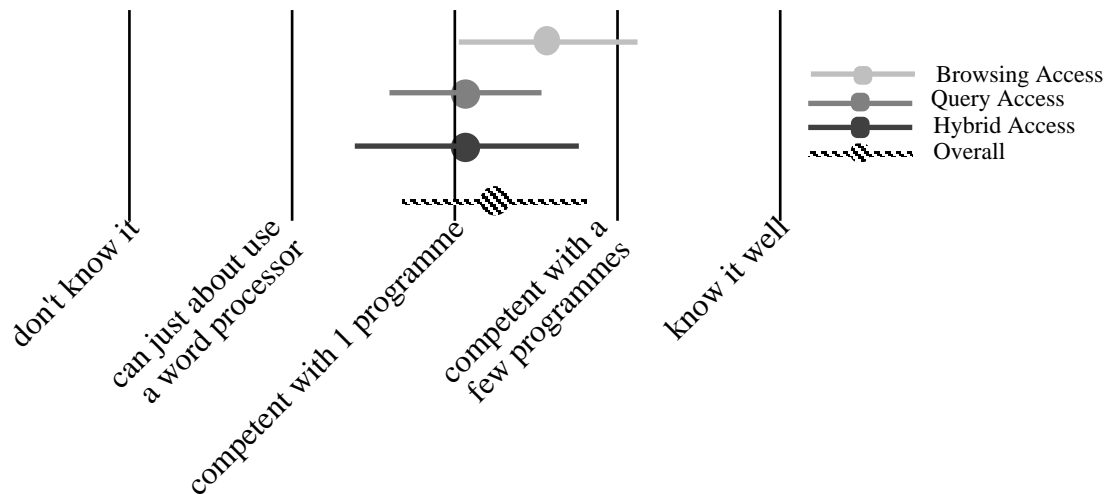




Figure 7.3: Graph of Macintosh experience

This shows that the average user claimed to be competent in one programme with a significant number competent in more than one program. 55% of users fell into the category of being able to use one program (either competently for one programme or manage to use a word processor), with the remainder of users being split between competence with a few programmes (28%) and knowing the Macintosh well (18%).

These two sets of figures on how well users assessed their knowledge of computing shows that the planned group of users had been achieved: users with enough knowledge to understand the instructions given and to use the computer without training but who could be considered as non-expert (or novice) users.

7.4.2 User Logs

As the tests were being conducted a log of the user's activities was recorded, this log noted all nodes visited, all commands given by the user, and all errors which were reported. A sample log is given below for a user starting to answer the first question of the test given as an example in section 7.11 (question number 188), the user's name is changed, but the log is not altered in any other respect. The user starts the query by asking about "Pelican Crossings", he then clicks the best match button () three times before deciding it is not doing anything, he then clicks the "go better" () button which results in an error (at 11:14:37). He then gives negative feedback twice on the best match (which is a picture of a staggered level crossing) and performs a feedback-based query which failed to match anything. Finally, he re-words his query and shows the next poorer document, rule 64, which he used to answer the question. The node column in the table shows the node which results from the action in the action section, i.e. the action moves to the node which is stated on the same line. A node number 0 indicates a comment, error message, or query.

<i>Time</i>	<i>Node</i>	<i>Comment or action</i>
10:58:03	0	Log for Mickey Mouse started.
10:58:03	0	Using query access on a Macintosh SE/30.
11:13:29	0	Query >Pelican Crossings
11:13:44	200	Show first relevant
11:14:13	200	Show first relevant
11:14:18	200	Show first relevant
11:14:37	0	There is no more relevant document than the current one.
11:14:53	200	(S)he hated that one
11:14:56	200	(S)he hated that one
11:15:29	0	Performing relevance feedback query
11:15:36	0	The document base contains no document relevant to your query. Please reword your query and try again.
11:16:53	0	Query >Pelican Crossings: these have a flashing amber signal, which foolows the red signal. What does this mean?
11:17:12	203	Show first relevant
11:17:36	203	Show first relevant
11:17:49	64	Show poorer relevant

7.4.3 Test Equipment

The tests were all carried out on the same Apple Macintosh SE/30 personal computer, this is a low to mid range Macintosh which uses a 16 Mhz Motorola 68030 main processor with a floating point maths co-processor. The machine did not have any special software or extra hardware to enhance performance. At the start of the tests the Macintosh was running version 6.0.5 of the Macintosh operating system, approximately one third of the way through the tests the system was upgraded to version 7.0. While this is a major upgrade to the Macintosh operating system there was almost no effect on the interface and appearance of *mmIR* (the software which users were tested with, see chapter 6 for further details). To ensure consistent performance for different users only the required software was running during tests (i.e. all background applications were quit before tests began), file sharing was disabled, and no changes were made to the software over the test period (despite various improvements which were suggested by the users' performance).

7.5 Analysis of Logs

The user logs were analysed in four stages: the users' logs were marked up to aid analysis, these marking were merged with the user logs and summarised, this data was then aggregated for each access method, and finally the results were produced by tabulating and graphing this data.

The users' logs were marked up to state the start of each query, in most cases this simply involved highlighting which query or *go home* command started a new query, but

in some cases *go home* commands were added where browsing-only users had moved on from one question to the next without returning home. Each question the users answered was marked with the question number for that questionnaire (i.e. the first question was number 1, the second 2, etc.), the query number as taken from Humphries (1987), the rule which the user gave as an answer, and the user's score. The score was given on a scale from 0 to 3, where 0 represents a solution which did not answer the question, 3 represents a solution which answered the question well, and 1 and 2 represent intermediate degrees of correctness. This scale of marking was adopted because strict adherence to the solutions given in Humphries was not possible since many questions have alternative solutions (either complete or partial). The alternative (partial) answers were given a score as users gave them, a check was then performed after all tests to insure consistent marking.

The markings made on the user logs (in paper form) were then merged with the user logs (in electronic form) to produce a summary of the user's activity. Software was written which would present the marker with every potential start of a query. For those that were actual query starts, the software processed the previous query (which was now known to be over) and requested the markings for the new query. The software analysed and tabulated, amongst other details which were not used in analysis, the question number (from 1 to 14), the time taken to complete each question, the number of nodes accessed (in total and after first seeing their eventual solution), the length of the initial query, and the user's score. The time taken for each query was calculated between the first node which was accessed and the user entering the last node before moving onto the next query. This removed the time taken for the user to read and understand the question, and to finally decide on and write down the solution. More significantly, it also excluded the time taken for the user to input a query and for that query to be processed (although it did include the time taken for relevance feedback-based queries or any textual queries which did not start a new question). Although it may have been beneficial to include these times it would have made the collection of data more complex and would have prevented the automatic timing methods, for example it would be impossible to automatically split how long a user took answering one query from how long (s)he took understanding the next query.

The log summaries were then aggregated for each access method to provide a single log summary for browsing, querying, and hybrid access. These aggregated summaries were finally graphed to produce the results shown in section 7.6.

7.6 Analysis Results

This section presents analysis of the results of the user tests. This section only presents results which can be calculated numerically from the user logs and questionnaires, more qualitative analysis is made in section 7.7. This section initially presents specific areas of analysis before presenting a general discussion of the results. All the graphs in this section, which show results for each query, have a dashed line between question 8 and question 9, this represents the separation between the two sessions.

7.6.1 Time Taken

The time taken by users to answer the questions is shown in figure 7.4, this measurement and the users' scores provide the most important measures of how well the users performed. The time taken shows how long it took the user to answer the question and so provides the most general statement for how easily the user could find a possible answer, the score shows how good this answer was. Figure 7.4 presents the results for each access method within a combined graph – showing mean values only. Separate graphs are shown in figure 7.5 together with error bars, these bars represent the standard deviation of the results (not the actual spread of results) – the bar extends from \bar{y} to $\bar{y} - \text{sd}(y)$ and to $\bar{y} + \text{sd}(y)$. The graphs represent the time taken on questions which were of approximately increasing difficulty during both sessions.

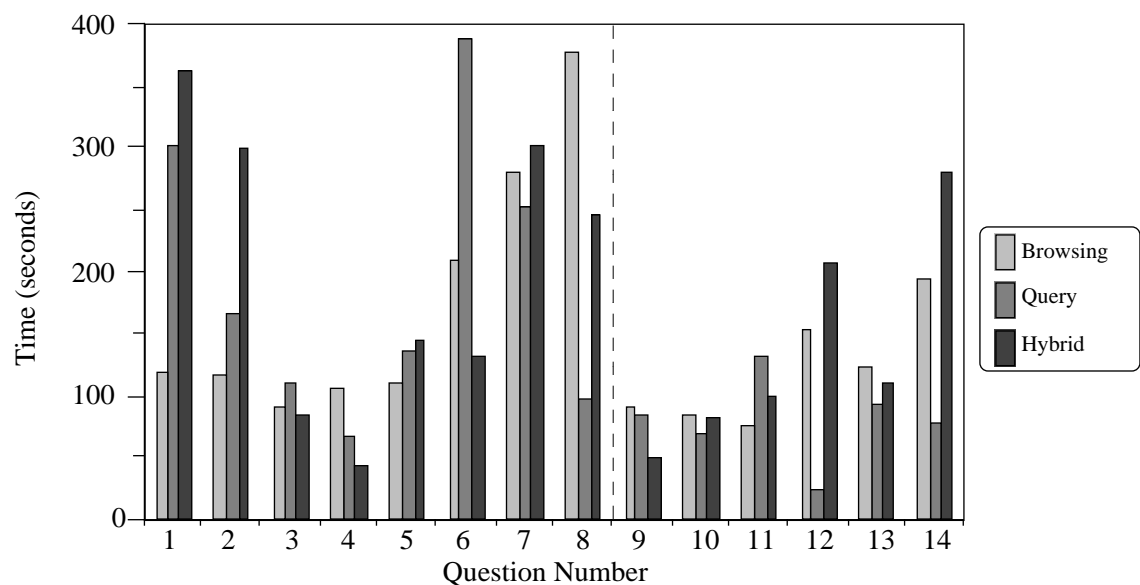


Figure 7.4: Time taken by users to answer each question on their questionnaire

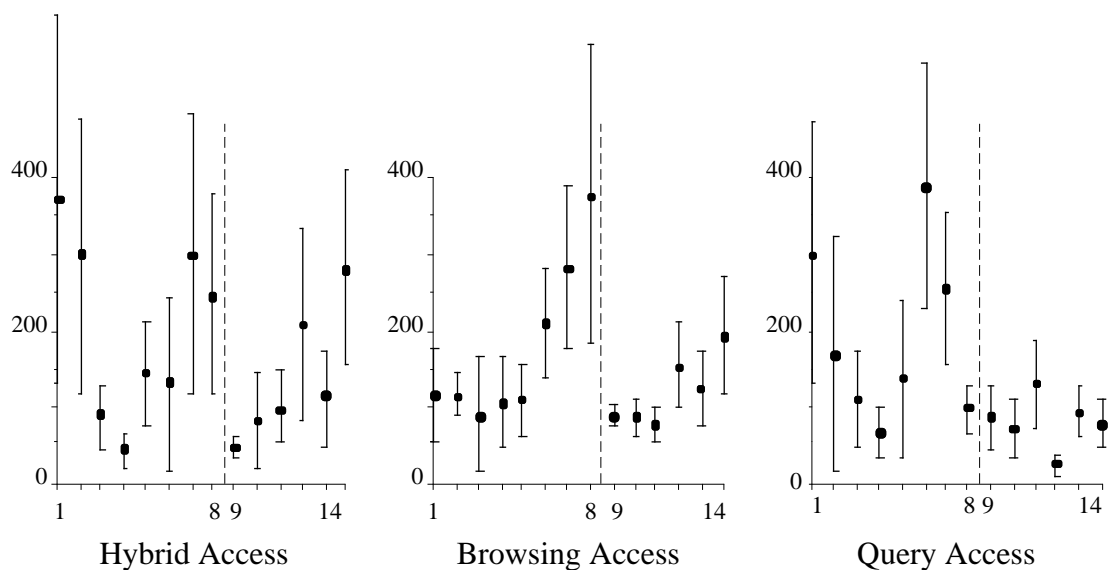


Figure 7.5: Individual time graphs

Although these graphs exclude the time taken for the user to enter their initial textual query, and for this to be processed, or for users to select which top level index to choose they do show many points which are of interest. The end of this sub-section discusses the effect that the thinking times has on the overall results.

Firstly, no access method is clearly better than any other method, and none is clearly poorer. Although a negative result this does show that for a relatively small document base (218 document nodes), one access method does not provide a great benefit over another.

Browsing-based access did provide the fastest access for novice users (approximately 3 times better than the other methods) for the first question. This lead was, however, rapidly eroded over the first three questions, which were of similar complexity, as the speed of browsing access only increased slightly but the performance of query and hybrid access increased considerably.

The middle section of the first session, between question 4 and 7 inclusive, showed none of the access methods to be particularly better than another method. Although browsing-based access did provide a more natural increase in time with complexity of question.

The final question of the first session shows a marked difference between the three access methods, with query based access providing the fastest times, browsing-based access the worst, and hybrid access almost half way between the two conventional access methods.

At the start of the second session all three access methods proved to be almost equally effective in answering simple questions posed to relative experts. None of the access methods appeared to suffer from the user forgetting aspects of the user interface. However, this may be an effect of using the system in a problem solving fashion as opposed to a real working environment. It should also be noted that the performance of the browsing-based access was not significantly better when the users were reasonably competent as when they were naive, adding further evidence to the claim that browsing-based access does provide easy access for naive users. The performance of the other access methods dropped considerably for easy tasks when compared to the start of the first session.

Considering the standard deviations, a measure of the spread of the users answering time, it can be seen that the hybrid access method has a much wider deviation, in general, than query and browsing only based access. This shows that the variation in time taken to answer questions is greater with the hybrid approach than with the other two approaches. With browsing-based access the standard deviation tended to increase, after the first few questions, with the difficulty of the questions. This is likely to be the result of some users managing to find the required answer rapidly, almost by accident, while other users are less fortunate and follow the wrong paths initially. Interestingly the standard deviations, and to a lesser extent, the mean values for query based access are very consistent over the

second session showing that the time taken for semi-expert users to answer questions is fairly consistent over the range in difficulty.

The late stages of the second session, questions 12 to 14 inclusive where the user may be considered as near-expert, showed an almost constant ranking of performance: query-based access fastest, followed by browsing-based access, and finally hybrid access. It is possible that the reasonably good performance of browsing access to hard questions could be caused by the users gaining a good knowledge of the document base structure. This view is not, however, be supported by the number of nodes which were accessed by the users. The graphs of time taken and nodes accessed are almost parallel, for browsing-based access, showing that users did not spend less time on each node as they became more experienced.

When considering questions 8 and 14, which were the same query for any given user, the only access method which showed a significant improvement in speed was browsing-only access. The mean access time for browsing only users fell from 376 seconds, for question 8, to 193 seconds, for question 14. This may support the claim that browsing-based access improves with time because the user learns the structure of the document base. This will be more apparent in these tests because of the relatively small document base.

Figure 7.6 gives a wider overview of the average time taken by users to complete questions. The graphs, one showing the mean time and one the standard distribution, are split into three sections: the overall average (and standard deviation) for the entire test, for the first session only, and for the second session.

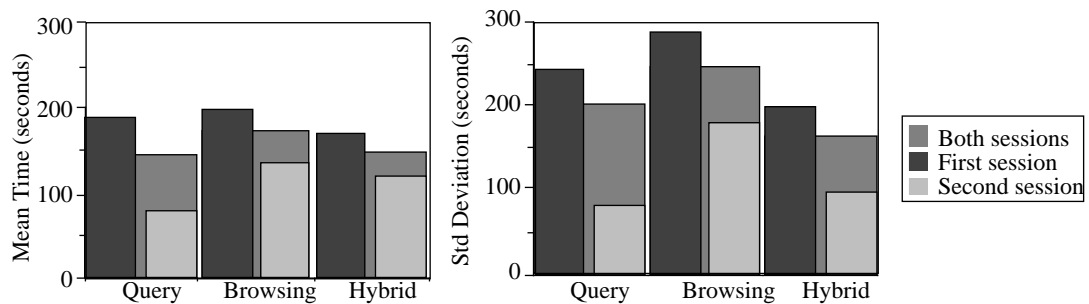


Figure 7.6: General results for time taken per question

These graphs show that all the access methods took approximately the same time to answer questions, with the exception of query based access in the second session. As these results give the overall score, but without the time taken for users to formulate their queries, the results are not very significant. They do, however, show that after the query has been formulated users of the hybrid access method performed noticeably poorer than users of query-only access. These results also show that the variation in times was slightly less for hybrid access during the first session with query access being next most consistent – however, the standard deviations are very large. The second session shows more stability for all access methods, with the largest improvement being for query based

access which was reasonably stable during the second session. The very high values of standard deviation for the first session suggests that the figures are dominated by learning effects and the differences in individual users. These differences are to be expected when users initially start using a system, the average figures given above do, however, give a feel for the overall level of performance as users learned the system. Scatter plots were drawn for all access methods and analysed to attempt to establish if any particular questions or users were consistently producing results considerably different than others. The plots did not show any such problems for questions – all questions asked at a particular stage of the tests proved to be of approximately the same complexity. Each access method did have a couple of users who were almost consistently poorer than others, but as they were not too much slower / poorer than other users they were not treated as special cases. No questions were shown to be consistently easier than any other for each position during a session. Similarly, no users were consistently much faster than others.

Effect of decision times

The mean time taken by users between each question was calculated for the first and second session and is shown in figure 7.7. These times are calculated by taking the time when the user first issued a command to the time when the user issued the last command for the entire session. The total taken recorded for users answering questions² was then subtracted from the total time. This figure states how long the user spent performing the tests which was not recorded earlier, this time was then divided by the number of questions in the session to give an average *thinking time* per question. The time taken formulating the first query or choosing which node to access first is not included in these figures (because this could not be calculated from the users' logs), but as the figures are averaged over the entire session the effects of this time would be small.

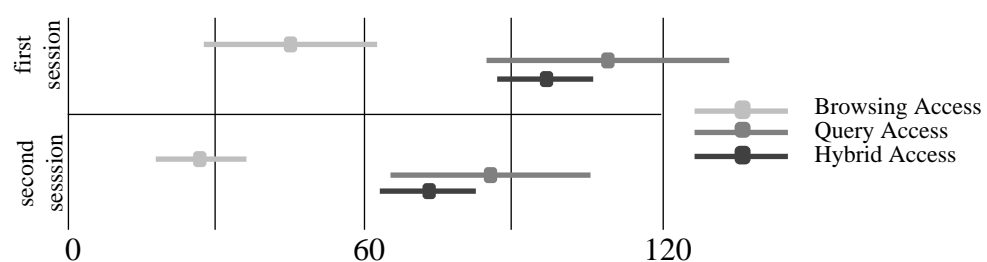


Figure 7.7: Average time between questions³

This shows that browsing-only users took considerably shorter to access the first node than other users. This is as expected since browsing-only users do not need to formulate or issue a query. Hybrid-users also took slightly less time to formulate their query than query-only users. This may show that hybrid users are less careful about the query formulation. The thinking times were added to the graphs of time taken to produce a

² as graphed earlier in this sub-section.

³ The standard deviations shown here are on a user by user basis and not a question by question basis.

question by question table of total time taken graph (figure 7.8), and a session by session graph (figure 7.9).

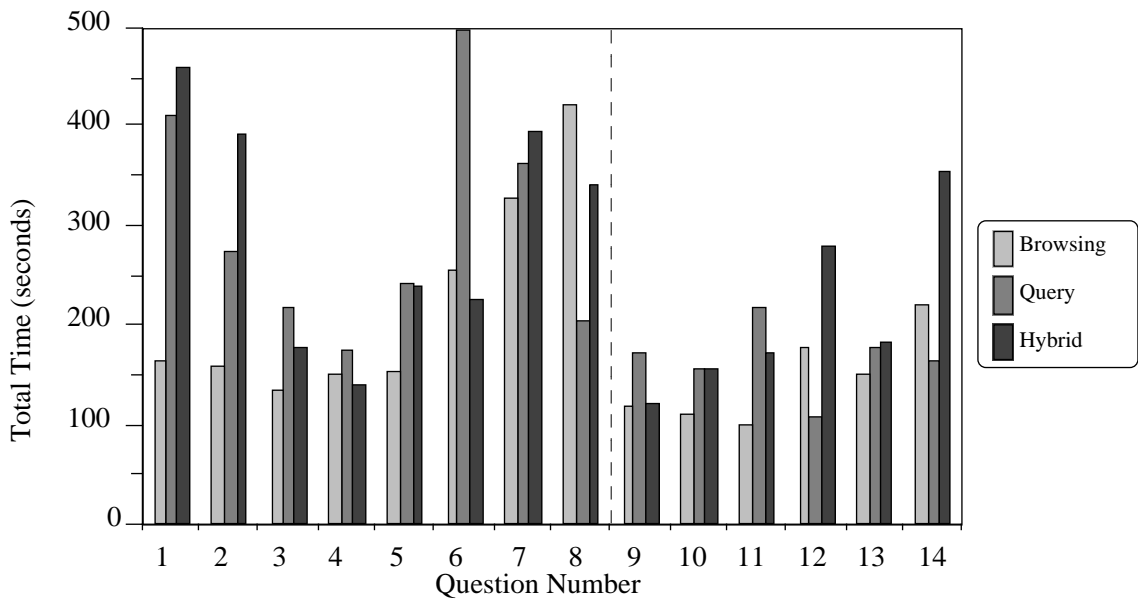


Figure 7.8 Time taken per question plus mean understanding time

Although the time taken by users to contemplate each question (and, where applicable, formulate a query), measured here, does not vary over each session, figure 7.8 does give an overall impression of how long users took to answer questions, and it shows no access method significantly better or poorer than any other. Figure 7.9 shows the mean total time taken by users to answer questions in both sessions. This is calculated by dividing the total time they took for the session by the number of questions in the session. As stated above, there is a small error factor introduced by the user's initial thinking time for the first question of both sessions not being recorded. This graph is a variation of figure 7.6 but with the thinking time added to each query. The standard deviations have been omitted because it is not possible to define this in terms of the thinking time.

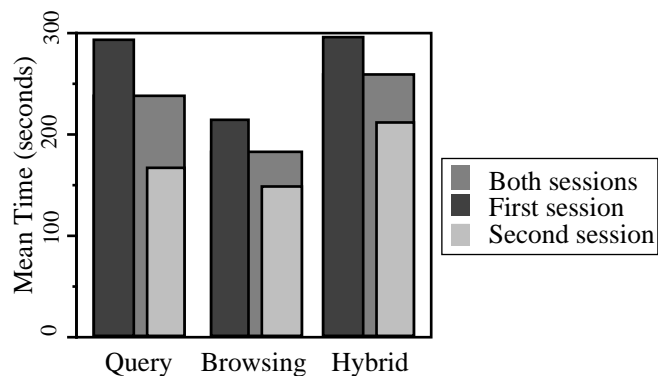


Figure 7.9: General results for time taken per question

Averaged over the entire test (■), browsing-based access showed a significant lead over the other access methods when the thinking time was taken into account. Query and

hybrid access users took approximately the same time to answer the questions. This compares with only a slight lead for query- and hybrid-based when the thinking time is not included.

During the first session (■) browsing-based access had a significant lead over the other access methods, which were very similar in speed. This compares to almost identical results between browsing and query only users in figure 7.6, which excludes the thinking time.

The second session (■) shows a considerable drop in the time taken by query-only users to the stage that they took almost the same time as browsing-based users. Users of the hybrid method were slower than the other users, but not greatly. This was due to hybrid users taking a reasonably long time to formulate their queries and then a reasonably long time to browse around the document base.

Summary of time analysis

Taking the thinking time into account browsing-based users were, overall, faster than other users. However, the lead was reduced drastically during the second session in which query only users were very similar in speed, and hybrid users were only slightly slower. In general query-only users took very little time looking around the matched document list while browsing-based users took very little time between searches. The hybrid access method may be inherently slower than the other access methods for novice users because they have to formulate a query (and this time was shown to be only slightly less than for query only users), and then browse around the document base (which took only slightly less time than browsing-only users). With a better user interface, as suggested later in this chapter, the speed of hybrid users would easily increase. An improved interface for hybrid-access is likely to bring the speed, at least, up to that of other access methods for semi-expert users and would give novice users performance between query-only and browsing-only access methods.

7.6.2 Quality of Solutions

As stated above the quality of solutions plays an important aspect in evaluating how well users performed with a particular access method. When the time taken does not vary considerably with each access method, the users accuracy becomes the only major numerical evaluation criteria. Figure 7.10 shows the average score for users' solutions, with 0 representing a solution which did not answer the question, 3 representing a solution which fully answered the question, and 1 and 2 providing intermediate levels of correctness.

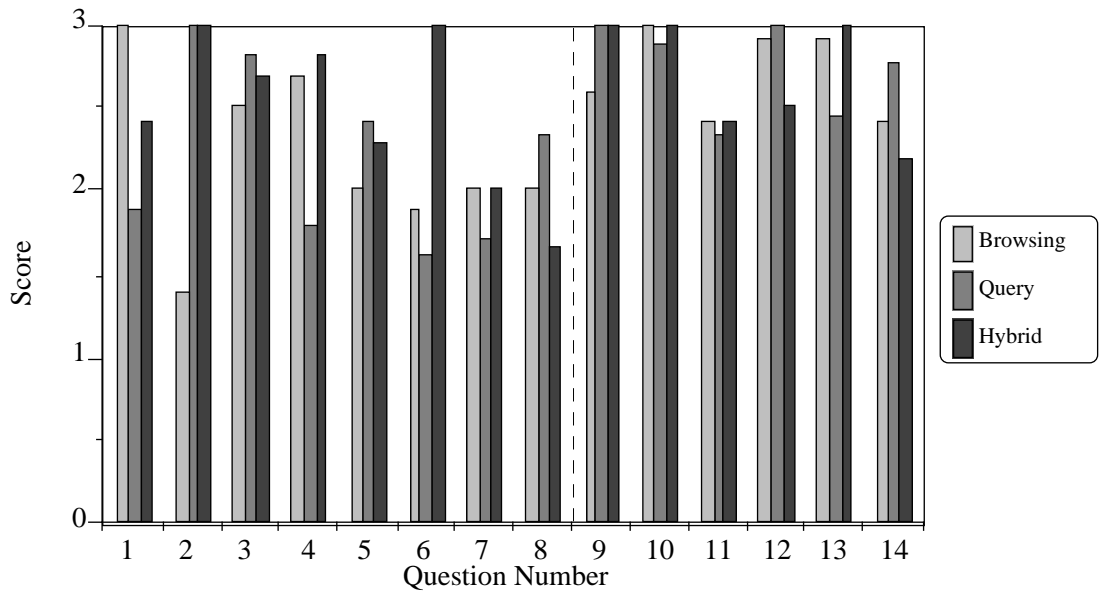


Figure 7.10: Average score which users solutions achieved.

The quality of solutions achieved with each access method was very similar, with no general trends in the mean figures except that the second session shows an improvement over the first. Figure 7.11 shows an overview of the scores achieved by users (together with the standard deviations).

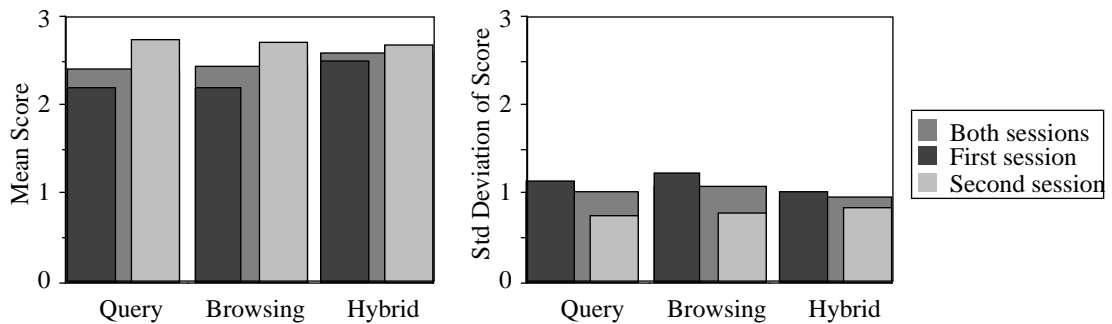


Figure 7.11: Overview of users' scores

The graphs in figure 7.11 show that the overall scores are very similar, with hybrid access being slightly better than other access methods during the first session but this is not too significant given the reasonably high standard deviations. The standard deviations also showed little difference between access methods, but there was a general improvement between session one and session two both in mean values and standard deviations.

The scores which users achieved did not show any strong correlation with the time it took users to reach the answers. However, there was a tendency for answers which scored zero to take slightly longer. Figure 7.12 shows the mean time for users to answer questions against the score which they achieved. The standard deviations for these values were very high and are not plotted. A graph of time against eventual score did not show any overall trends.

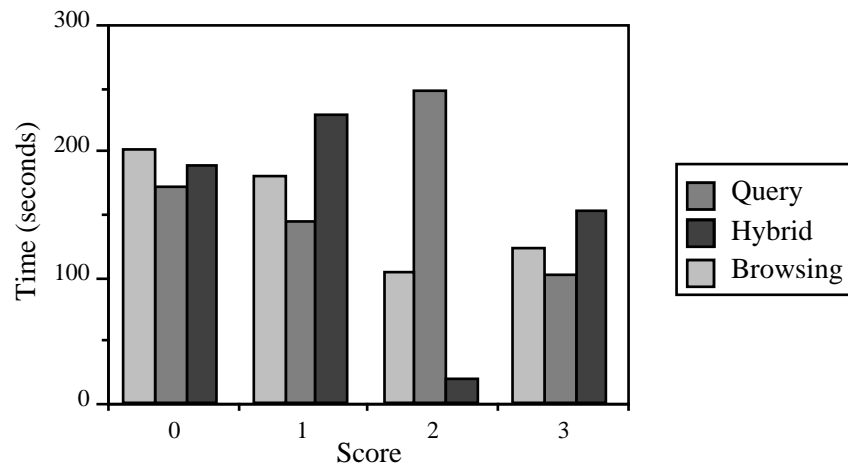


Figure 7.12: Users' scores against time taken

7.6.3 Nodes Accessed

Figure 7.13 shows the average number of nodes which were accessed by users in order to answer each question. Although the overall time taken to answer a question is the most important method for measuring the user's speed of answering, the number of nodes which were accessed on route provides an alternative view of the users' activities.

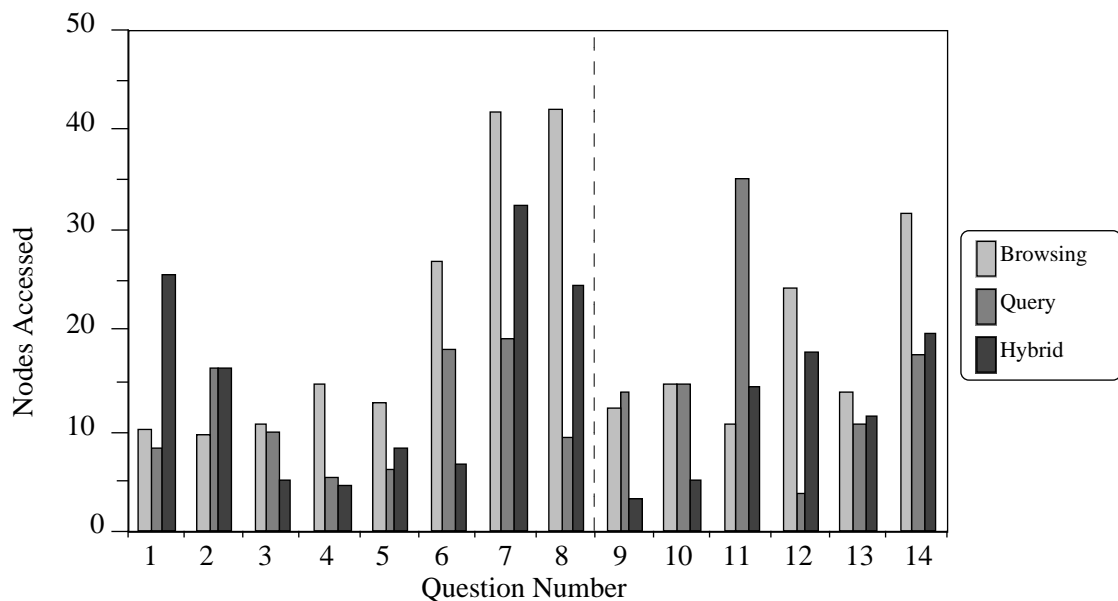


Figure 7.13: Number of nodes accessed by users in order to answer each question

Once again neither access method provided a significant difference from the other methods. At the start of the first session users of the hybrid method typically accessed more nodes than the other two access methods. This was partly due to their confusion over which navigation controls (arrows) to use, and therefore examining the effects of the controls. Browsing and query-only access methods produced almost an identical number of nodes accessed for the first question. Over the first three questions, which were of very

similar complexity, the number of nodes accessed dropped considerably for hybrid access, reinforcing the belief that the initial high value was due to exploration, while browsing-based access stayed almost constant and query-based access had no well-defined path, although a drop in nodes visited was shown between question 2 and 4 inclusive.

During the middle of the first session the number of nodes accessed grew almost constantly for all access methods. At the end of the first session there was a considerable difference in the number of nodes accessed by each access method, combined with the time taken to answer question 8 this shows that the difficulty involved in answering this question is different for each access method with query access being easiest and browsing only access the hardest.

At the start of the second session the number of nodes accessed was very similar for browsing and query access, but considerably lower for hybrid-based access. Over the remainder of the second session hybrid access showed an almost continuous increase in the number of nodes visited. Browsing-based access shows a general increase in the number of nodes taken to answer each question over the second session. Query-based access showed no overall trend to end with approximately the same number of nodes as were accessed for the first question of session 2.

Comparing the behaviour of users between question 8 and 14, which were the same question for each user, one can see that the number of nodes accessed by browsing users dropped noticeably, while the number for hybrid access decreased slightly and query only access actually increased slightly. The overall effect was to narrow the gap in number of nodes accessed to answer the same, difficult, question. Although there is a minimum number of nodes which must be accessed to answer these questions these figures are much lower than those achieved (approximately 5 nodes for query and hybrid access, and 10 nodes for browsing access). A comparison can be made with the expert figures given in section 7.6.9, to see how close the figures were to a realistic minimum.

7.6.4 User Confidence

The users' confidence in their answers was assessed, initially, by considering how many questions the users felt they had answered correctly. The users were asked to split their answers into six categories: definitely correct (A), probably correct (B), maybe correct (C), probably wrong (D), definitely wrong (E), and not answered (F). A single score was calculated for user performance based on the following algorithm, in which the letters A–F represent the number of questions which the user considered fell into that category:

$$\text{expected score} = \frac{1}{n}(4A + 3B + 2C + D)$$

where n is the number of questions the user was asked to answer.

This provides a value between zero and one which can be compared with the mean score which the user achieved. No *points* were given for questions which were not answered or which were considered definitely wrong (E and F). The results are shown in figure 7.14.

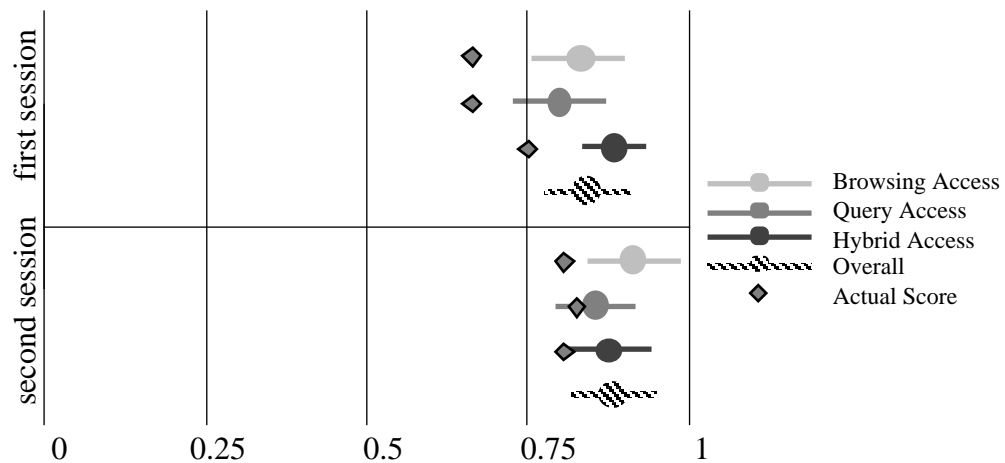


Figure 7.14: Graph of users' expectation of answer correctness

Overall users were reasonably confident that they had answered the questions well, users of all access methods had similar expectations except for query only access which had a slightly poorer expectation value for both sessions. Interesting results come from comparing the difference between the expected performance for the first and second sessions and in considering the difference between expected and actual performance (mean values shown as diamond bullets – scaled from 0 to 3 down to 0 to 1).

Confidence increased for query and browsing users but stayed almost constant for hybrid based access, though hybrid access did get a reasonably high expectation for both sessions. Browsing based access confidence rose considerably for the second session to a value very near perfect, this may be associated with the users having greater knowledge of the document base.

Considering the difference between expected score and actual score, the first session showed a considerable difference between the two values. This is not surprising considering the relatively low scores achieved in the first session. The difference between actual score and expected was very similar for all access methods during the first session. However, this difference did vary during the second session. With query-based users having approximately the same expectations as actual score with others still being lower than achieved. In all cases the difference between expected and actual scores was reduced to some extent.

One of the major issues affecting the user's confidence in a traditional free text retrieval system occurs when the user fails to have their needs satisfied, and feels that the document base contains the information but that (s)he cannot find it. The equivalent within these user tests occurred when a user could not answer a specific question, under these circumstances the users were asked to state how confident they were that the document base did not contain any information which would answer the question. There were only 13 occasions when the user could not answer a question, roughly 3% of the questions asked – these occurred with 8 users, three users did not answer 2 questions and two users failed to answer 3 questions. Figure 7.15 shows how confident the users were that the

document base did not contain the answer, the users answered this question on a scale from 0, for not at all confident, to 6, for very confident there is nothing relevant to the query.

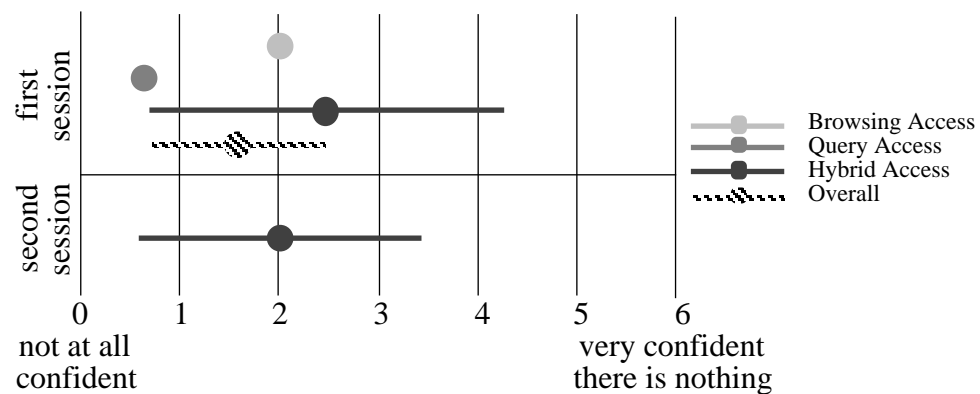


Figure 7.15: Graph of user confidence for non-answered solutions

Although these results are not very significant due to the few occurrences (see table 7.1), it would appear to show that the users of query-based access were less sure that the document base did not contain anything that would answer the query. This may be a result of the restrictive browsing facilities which they are given. The document base may also affect these results, as it is well known and expected to be able to answer all questions on driving.

Overall	Hybrid Access	Browsing Access	Query Access
11 (8)	2 (2)	4 (3)	5 (2)
2 (2)	2 (2)	0 (0)	0 (0)

Table 7.1: Number of failures to answer a question (figure in brackets show number of users involved)

An alternative view of how confident users felt while using the retrieval engine can be achieved by considering the number of nodes which the user accessed after visiting the node which they eventually gave as an answer (*check nodes*). Figure 7.16 shows the number of check nodes which the user visited, again the results are not very significant due to the very high standard deviations, but users of query based access do appear to access more check nodes than other users. A full graph showing the number of check nodes against question number does not show any general trends, but it does show that the mean number of check nodes does not decrease significantly as users become more competent (see figure 7.17) and that query based access does incur more check nodes for the majority questions.

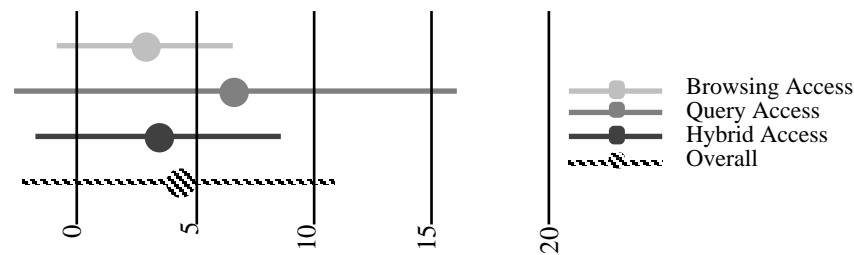


Figure 7.16: Graph of nodes accessed after visiting answer node

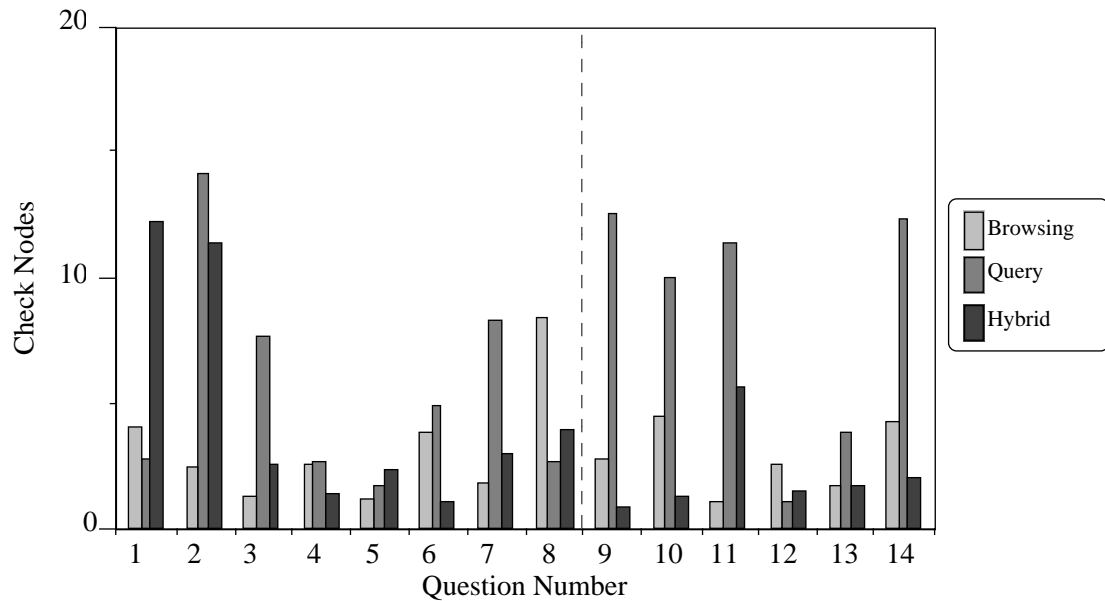


Figure 7.17: Average number of nodes accessed after visiting answer node

Overall, the confidence that query-only users have in their answers appears to be slightly lower than other access methods. No significant difference between browsing and hybrid-access users confidence was found.

7.6.5 User Speed Expectations

To attempt to establish how fast the users found their access method they were asked three questions: how many steps did it take to find the answers (figure 7.18), how fast they found the computer's response (figure 7.19), and finally whether they felt they could do better with a computer or the paper based Highway Code (figure 7.20).

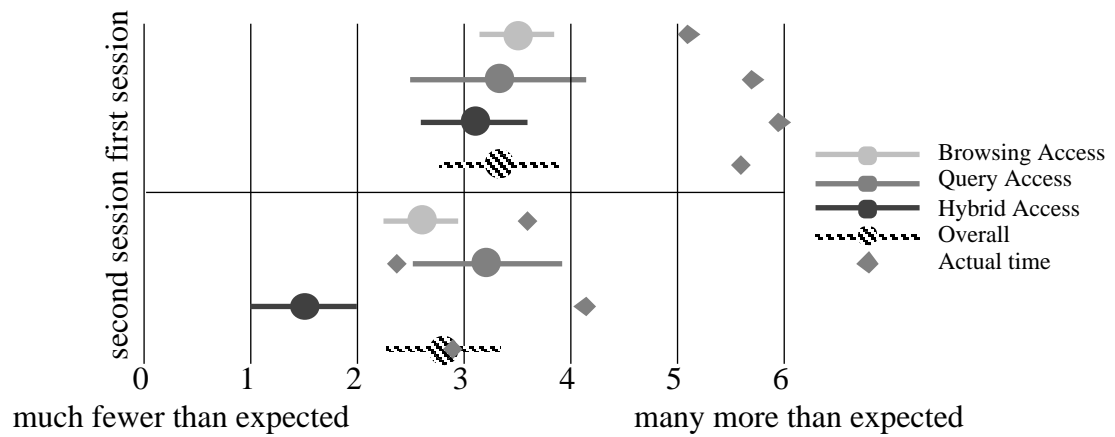


Figure 7.18: Graph of how many steps users felt they took to answer the questions.

Figure 7.18 shows how many steps users felt it took to answer each question, together with how long it actually took them (on a scale from 0 to 6 where 6 is the longest mean time).

The graph shows that, overall, users felt that they took approximately the expected number of steps to find the answers. Only hybrid users during the second session felt significantly different, as they felt they took fewer steps than expected; this may be related to the long time taken by users during the first session.

The difference between how many steps users felt they took and the time they actually took was similar for all access methods during the first session. The difference was greatest for hybrid users who felt they had taken the fewest number of steps but had taken the longest to answer the questions. This may, however, be a result of hybrid access having a low average number of nodes visited. During the second session hybrid users also felt that they had taken the fewest number of steps but had actually taken the longest time, whereas query-only users felt they had taken the most steps but had actually taken the shortest time. It is also interesting to note that query-based users feeling was almost constant despite a halving in the time taken to answer questions. While far from conclusive these results do tend to show that a user's feeling for how *many steps* it took to answer a question does not relate directly to time and the relationship varies with each access method. It also shows a slightly greater confidence for hybrid users.

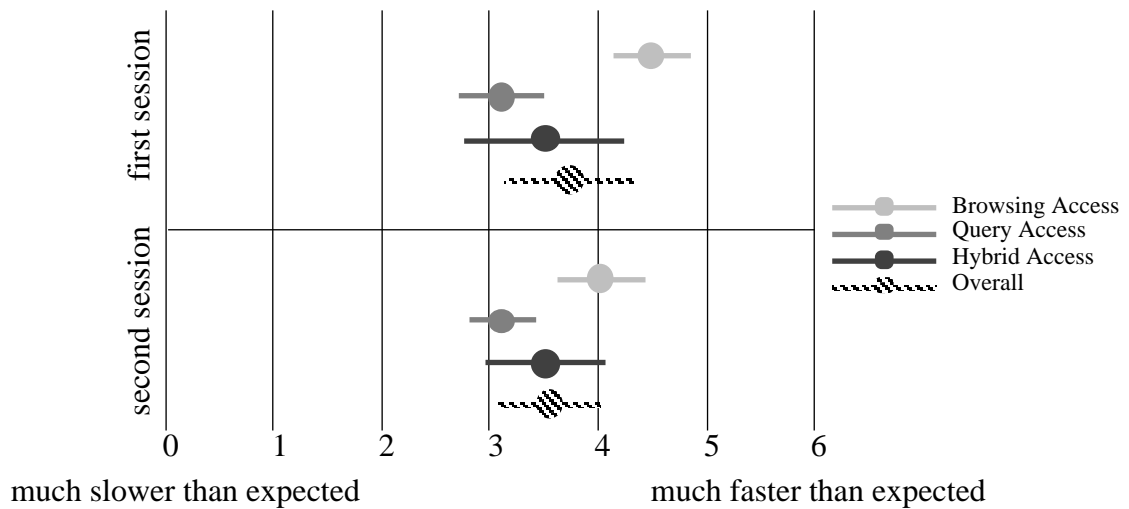


Figure 7.19: Graph of how users rated the computer's response time.

Figure 7.19 shows that, overall, users found the reaction time slightly faster than expected with browsing-based users finding it quite a bit faster than expected. This is quite significant because of the reasonably small difference between query and browsing access (especially during the second session) despite it taking approximately 30 seconds to execute a query but at most fifteen seconds to browse to a node (and often under two seconds). Users partly adjusted their expectations of speed with respect to how much work they felt the system was performing, this may cause problems for the hybrid document base model if it is presented through a browsing-only interface. The calculation of link strengths, to filter out irrelevant links, may not be possible within the short time which users expect link following should take.

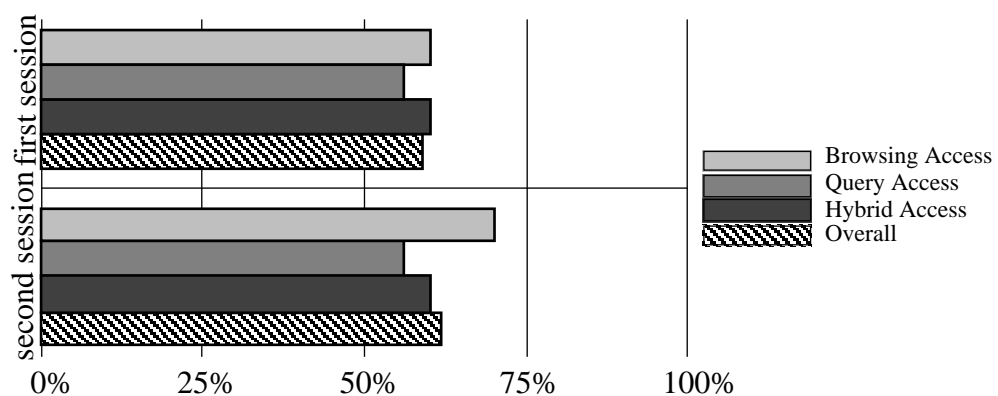


Figure 7.20: Graph of how many users felt they could do better with a computer based Highway Code

Figure 7.20 shows an overall approximate halfway split between users feeling they could perform better with a computer or paper based Highway Code. Browsing-based users appeared to prefer the computer based Highway Code slightly more than other users but this is a very small difference. It should be noted that the errors on this graph are reasonably high. In particular, the higher result in the second session for browsing-only

access was due to a single user changing his view. However, the graph does give an impression for how many users felt they could perform better with the computer

7.6.6 User's Surprise at Results

After the users had their solutions marked they were asked to state how surprised they were at the results. Figure 7.21 shows the results of this question.

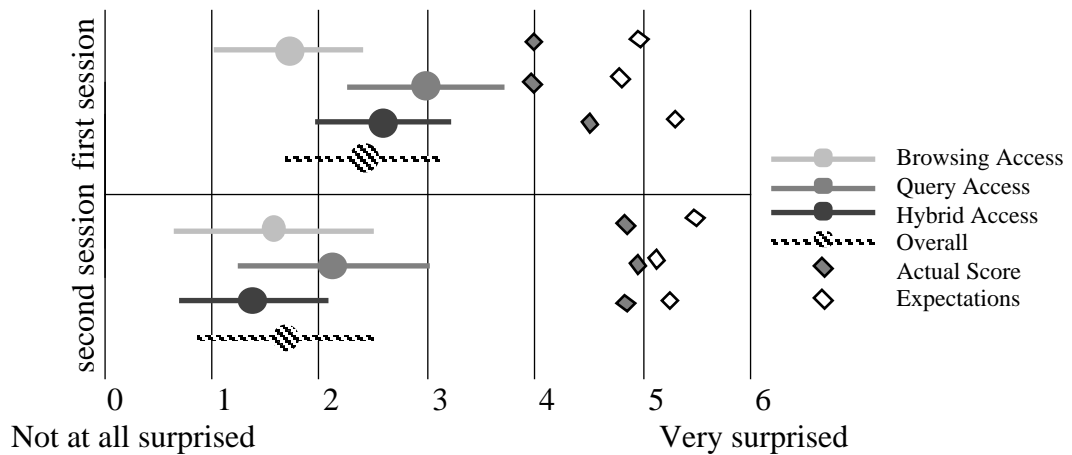


Figure 7.21: Graph of user's surprise at their results

The results show that, in general, users were not very surprised at their results, and that they were less surprised after the second session. The browsing-based users showed significantly less surprise at their results than other users during the first session despite all access methods having approximately the same difference between actual score and expectations. There was little difference between the surprise users expressed after the second session.

7.6.7 Query Length

Although not directly connected with the comparison of each access method the average length of an initial query to the retrieval engine was considered. The initial query was taken as the first query which was issued at the start of each question; re-formulations of the query, and feedback-based queries were not considered. Figure 7.22 shows a graph of the length of the original query in words, no plot is made for browsing-only access which did not involve querying.

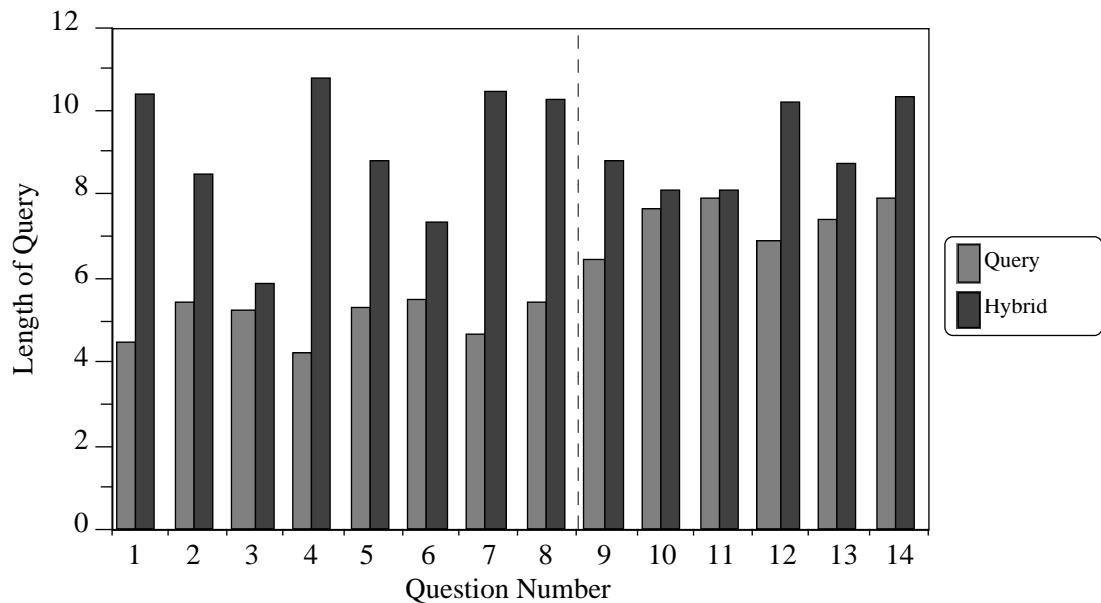


Figure 7.22: Length of first query for each question (in words)

This graph shows a general increase in the length of the initial query for query only access (from approximately 4.5 words for the first query to 7.9 for the last). The graph of query length for hybrid-based access was considerably more unsettled, but always remained higher than for query only access. The result for query-only access is highly significant as it shows that, when users are presented with a retrieval system which performs better when many terms are used in the query, users will actually notice this increase in performance and provide longer queries. It should be noted that the window into which users typed their queries was very large and could easily accommodate many terms, this may have a significant impact on persuading users to input longer queries. When combined with the shorter time which hybrid users took in formulating queries, this graph may show that query-only users took more care, and time, formulating their queries than hybrid users. This may be a result of the lower reliance on the query which hybrid users have, and hence the irregularity of this graph.

7.6.8 History Usage

The history commands were mainly developed as a basic navigational tool for browsing only users, but were made available to all users. Figure 7.23 shows the mean number of times history commands (go back, go forward, and go back to last relevant) issued by users in total (i.e. between both sessions).

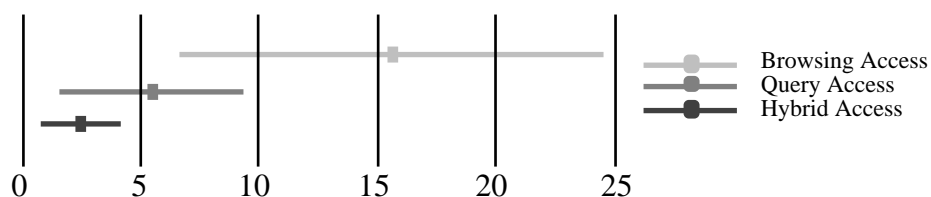


Figure 7.23: Uses of history commands per user

This shows that browsing-only users did use the history command considerably more often than users of query-only and hybrid access methods. This is reasonable evidence that browsing does, indeed, require a history mechanism more than other access methods, although the very high standard deviation shows that there is a wide spread in the usage. Interestingly, despite being given a more complex interface, users of hybrid access used the history commands less than users of query only access. This may be a result of the users struggling to understand the basic access methods without using additional ones, or of users being able to recover by use of other access methods.

7.6.9 Expert Trial

To provide a realistic definition of how well users can be expected to perform with the document base, the author ran through a test session for each query. These results, although based on a single run through for each access method, give a feel for how a user who knows all access methods, and the document base, well can perform.

Figure 7.24 shows the total time taken by the author to answer questions in each access method. This graph includes the *thinking time* described at the end of session 7.6.1.

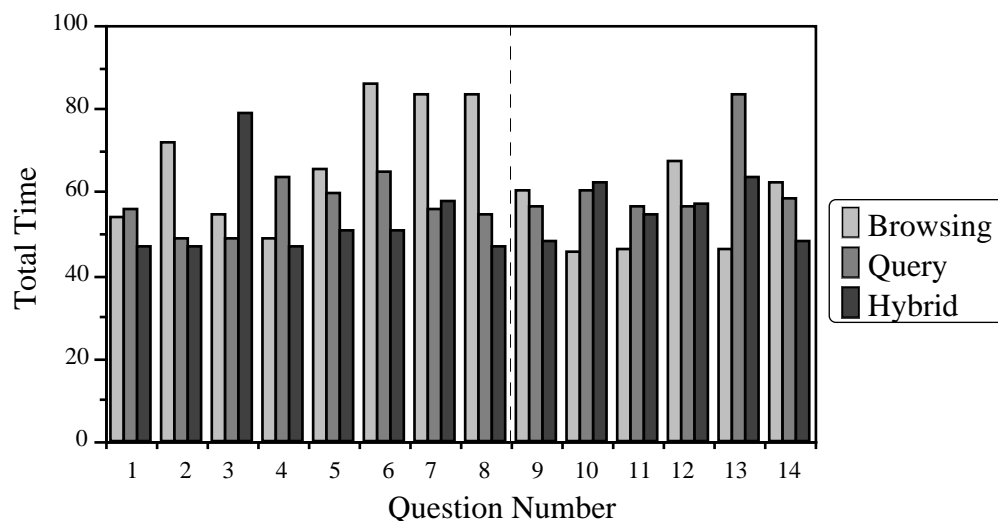


Figure 7.24: Total time taken by expert user.

This graph shows no overall trends in the time taken to answer questions, except the continuous increase of browsing-access during the first session. Both the overall lack of trends and the smooth increase for browsing-access may be a consequence of the expert user never having problems answering the given questions. The mean time taken per session, and overall, is shown in figure 7.25 together with standard deviations.

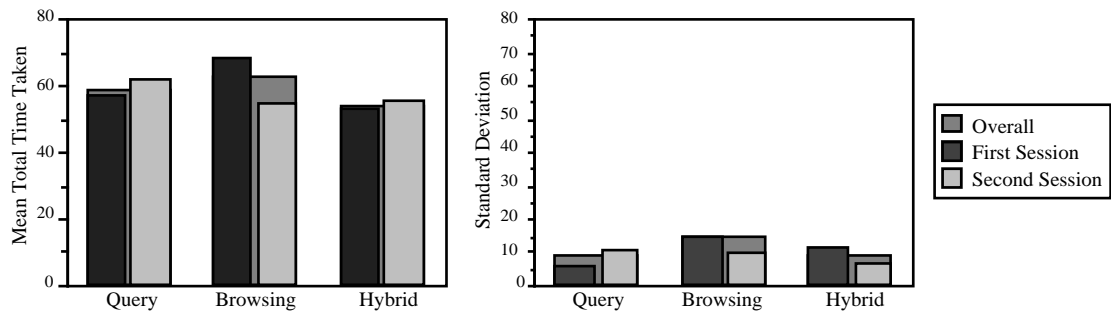


Figure 7.25: Mean total time taken by expert user.

The mean time graph shows no overall advantage for any access method. The standard deviations are also very low (the standard deviations do not take into account variations in thinking time). These tests show that no group of users approached a true expert user level of speed, and that all access methods were approximately the same distance from achieving the maximum performance. For the second session the test users took approximately 2.5 times longer than the expert user (the realistic minimum time).

Because the expert user achieved a perfect score for all access methods, time is the only major criteria in considering the quality of each access method for the expert tests.

The number of nodes accessed provides another factor which is of interest in comparing with values achieved by real users. Figure 7.26 shows the number of nodes accessed by the expert user.

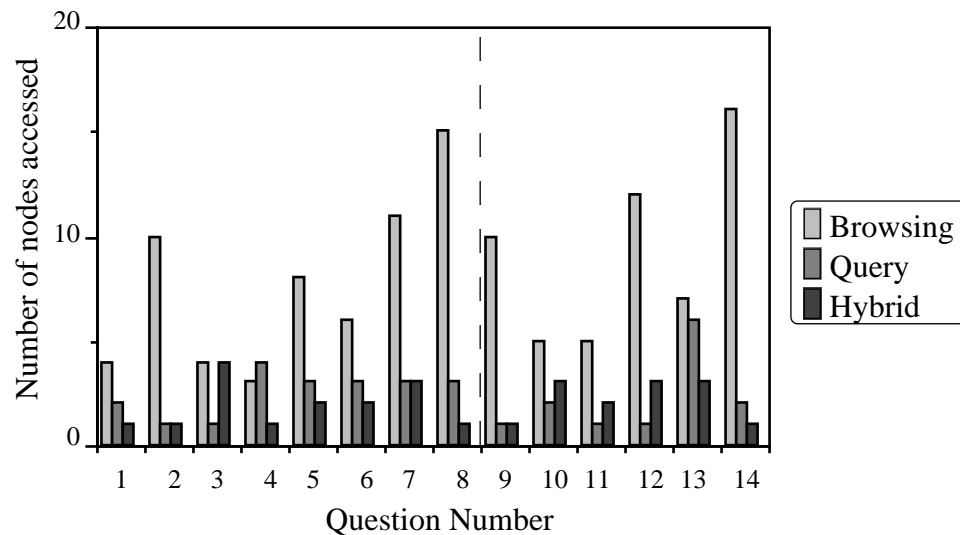


Figure 7.26: Number of nodes accessed by expert user.

This graph shows no general trend for number of nodes accessed for any access method, with the exception of browsing-access during the first session which shows a reasonably smooth increase. It does, however, clearly show that browsing based access requires many more nodes to be accessed per question than other access methods. Figure 7.27 shows the number of nodes accessed averaged over the sessions.

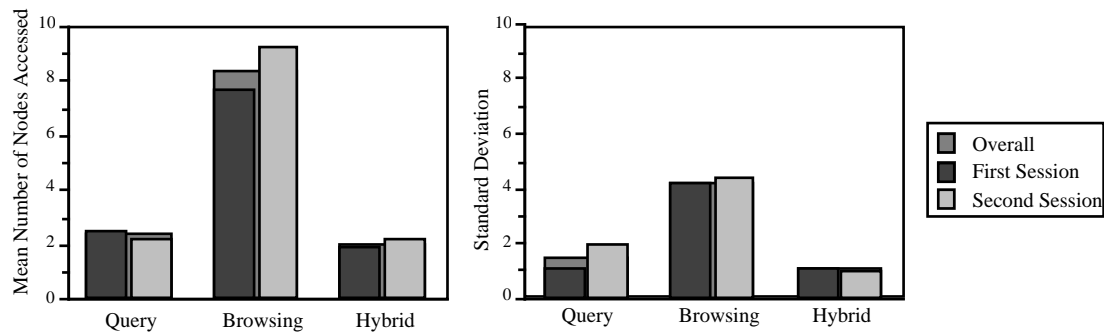


Figure 7.27: Mean number of nodes accessed by expert user.

These figure show that the expert user accessed very few nodes in order to answer questions with query-only or hybrid access, and relatively few for browsing-only access. The differences between the number of nodes visited between the expert user and the test users is less than the difference for total time taken. This shows that the most important reason for test users being slower is the time taken to decide which route to take. Interestingly over the second session, in general, users of browsing based access were closer to the minimum number of nodes than users of the other access methods. At the start of the session hybrid access was also very close to the minimum number of nodes accessed. However, the difference between access methods reduced. Figure 7.28 shows the difference between the actual number of nodes taken by test users and the mean realistic minimum number of nodes, per question, for the second session.

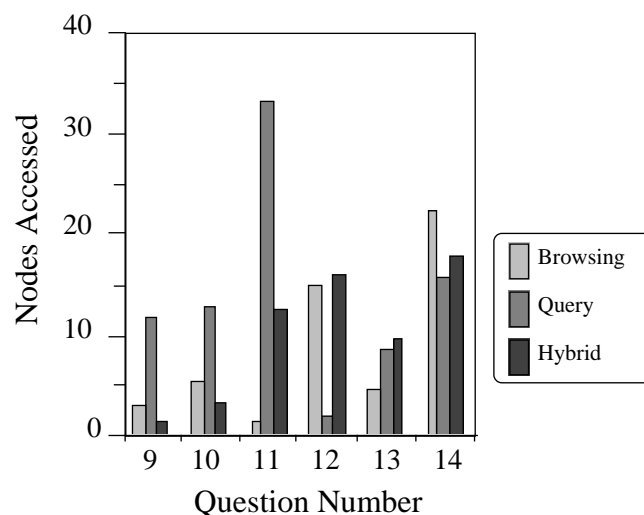


Figure 7.28: Difference between number of nodes accessed by expert and test users

The last graph in this section, figure 7.29, shows the length of question which the expert user issued.

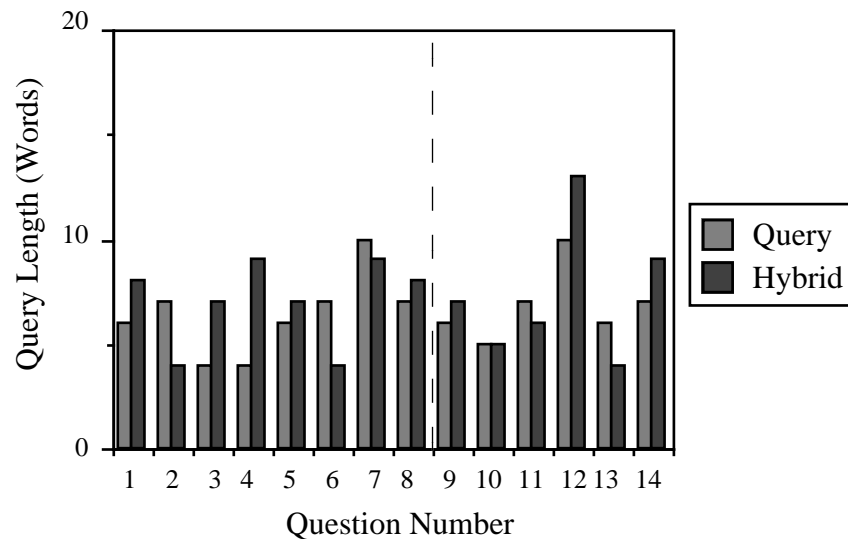


Figure 7.29 Length of expert user's queries

The number of words in the expert user's questions was very similar to that of hybrid users during the main tests, and that which query-based users were creating at the end of the second session.

7.7 Observation Results

This section discusses many issues of the users' interaction which did not come through in the more formal analysis of user logs but from observation and from the understanding checks which were issued at the end of the second session.

7.7.1 Relevance Feedback

Users found relevance feedback quite difficult to use. Only 42% of users (of hybrid and query access) learned to use it correctly – i.e. knew that they should give feedback and then issue a relevance based query, and only 26% used feedback more than occasionally. Many users clicked the feedback icons (happy and sad men) and were put off by the lack of feedback, the icon simply highlighted and then returned to normal with no apparent change in the status of the windows. Admittedly this was a design failure in that the node should display the current relevance value (marked relevant, irrelevant, or not marked) and this would provide the user with interface feedback that the action had been acknowledged. The users did, however, also appear to expect the system to react immediately to their feedback and start a new query, the provision of interface feedback would reduce this problem by at least reassuring the user that the action was accepted and the user would probably continue searching for some method to further use the information. On the understanding test at the end of the second session, 47% of users stated correctly that feedback was taken into account at the next feedback based query. This shows that approximately half the users of query and hybrid feedback did not understand the basic operation of feedback (provide feedback and then issue a feedback query), despite this

being outlined on their introductory guide and being a basic technique of information retrieval.

Negative feedback posed a great problem, with only a couple of users managing to use it correctly. Most users rarely provided negative feedback, but when they did they tended to give it on nodes which almost matched the query, but not quite. They simply skipped past nodes which were completely irrelevant (e.g. users would mark a node discussing the right hand lane of a two lane motorway, as opposed to a three lane motorway, as irrelevant, but skip past a node discussing cycling on roundabouts). This is exactly the opposite behaviour to that which gives the best results for negative feedback. When the user is expected to comment on documents which are irrelevant as opposed to partially relevant. The experiments conducted by Ide and Salton (described in section 2.2.4) reduced this problem by forcing users to give feedback on every node, under these circumstances the user was forced to mark completely irrelevant nodes as irrelevant. The overhead of having to provide feedback on every node may, however, be too large for a working retrieval engine and will strongly restrict the style of user interface which is used. The reluctance of users to give negative feedback may be due to the cognitive switch which must take place for users to provide negative feedback. While browsing through the matched documents (and, where possible, their neighbours), and while giving positive feedback, users consider that they are getting closer to the eventual solution. However, when they give negative feedback they may consider this a deviation from the path to a solution and are, therefore, not as likely to use it.

Many users tended to provide relevance feedback **after** they had written the answer down and just before moving on to the next question. These users obviously felt that their feedback would, in some sense, aid future retrievals. This would give hope to research into machine learning through the use of user feedback. Users would appear to be very willing to provide feedback on the best actual match and state that “this is what I was looking for”.

7.7.2 Disorientation

There were two occasions in which users tended to become disoriented: when following links to other sections of the Highway Code, and when browsing off the matched document list. When browsing-based users followed links, which took them from one section to another section of the Highway Code, they typically did not realise that the *go up* (go to parent) command would take them to a different index from the last one they viewed. The user typically had to look at a couple of nodes in the new section before realising that this area was not relevant to their query. The use of navigational aids (e.g. maps) would help to reduce this problem as the users would notice that their location in the map has changed from the section they are interested in to another section. This specific problem is likely to be more noticeable in document bases, like the one used here, which have a strong hierarchical component. The problems might not be as simple in more

general graph hypermedia documents but the user is still likely to be disoriented when (s)he follows a link to a distant part of the document base.

User⁴ disorientation after browsing off the matched document list, was envisaged as a major problem with the current implementation of the matched document list. In practice the problem did not occur as frequently as was expected. Users tended to only look to the sides of the matched document list and then return to the node where they started browsing from and continue down the list. Section 7.8.1 gives an alternative to the hidden matched node list which is expected to remove the problem of users browsing off and not returning to the correct position in the matched document list. When asked at the end of the second session to describe where, on the matched document list, one would be after moving right from the second best match, only 2 users confidently said that one could not tell what position in the list one would be at. The remaining users were equally split amongst the other options.

7.7.3 Query Style

The style of users' queries presented an interesting comparison with how users described the workings of the retrieval engine. The vast majority of queries were greater than one word long. A significant number of queries even went to the stage of correct capitalisation and punctuation - although many of these were almost direct quotes from the questionnaire. Users did, on the whole, interpret the question into a shorter query which was a well formed sentence. Although the users were issuing queries in natural language, when asked to describe how the retrieval engine worked, 89% of users described the process in terms of term-based retrieval. This difference would appear to show that, although realising that the system was working on a term-based model, users found it more natural to express their queries in natural language as opposed to a list of terms. This is, indeed, the way that I tend to use the retrieval engine as the benefit of providing many lightly weighted words appears to be of considerably greater than that achieved by attempting to use a couple of highly weighted terms.

7.7.4 Link Types

Although many users could make nearly correct guesses at the meaning of the various link types (*similar to no X*, *Rule no X*, *Picture no. X*, and *Index no. X*), only 44% of users had realised the difference between *Rule No. X* and *Index No. X* while using the system. This meant that the majority of users did not know whether following a link from an index node would take them to a rule or to another index document. When viewing a rule only 22% of users noticed and roughly understood the difference between *Similar to no X* (nearest neighbour) links and *Rule no X* (direct link within paper based Highway Code). These two low figures may indicate that displaying the type of link to users is not of help

⁴ Users of the hybrid access model, in which they could issue queries **and** browse through the document base.

to the majority of users. However, as the provision did not appear to introduce a complexity overhead it would be worthwhile displaying the link type if it can be naturally done within the chosen interface style.

7.8 Suggested Improvements

This section presents a few suggestions which would lead to an improved user interface. It concentrates on the hybrid user interfaces (i.e. those which provide both query and browsing facilities) as these most naturally exploit the benefits of an underlying hybrid document base. User testing has also shown a basic hybrid interface to be as effective as browsing-only or query only interfaces.

7.8.1 Integration of Queries and Browsing

In the prototype application the results of a query were presented as an implicit list of matched nodes which the user could browse through using navigation commands (up and down arrows). Although simple to implement and reasonably effective within a query-only interface this approach proved too simplistic when used in a hybrid query and browsing system. The single issue of having to return to the last position in the list of matched nodes, before continuing to scan down the list, caused many problems for hybrid-users during the user tests. An example of the worst scenario occurs when the document base contains the structure shown figure 7.30: a list of matched nodes and a path which leads through non-matching nodes back to the matched list but at a different location.

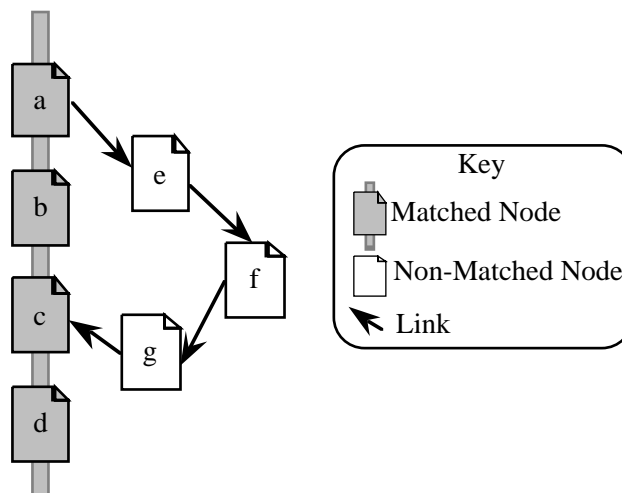


Figure 7.30: Sample network from prototype application

In this example the user may follow links from *a* through *e*, *f*, and *g* to node *c*, at this point the designer of the retrieval interface is faced with a problem: should node *c* be presented as a matching or non-matching node? If it is presented as a matching node, as was done in *mmIR*, the user is at great risk of wrongly deciding to continue scanning down the matched node list from this point, resulting in the user completely missing node *b*. The alternative approach, of displaying node *c* as if it were not a matched node, has the

advantage that the user would return to the last matched node, *a*, before continuing to search down the list. This approach does, however, have the problem of sometimes displaying node *c* as a matched node (e.g. after scanning down from *b*) and sometimes as a non-matched node (e.g. when browsing from *g*). This inconsistency may lead to problems with users' understanding the structure of the document base.

A solution to this problem lies with a technique of matched list display which is becoming more common in mouse-based information retrieval systems (e.g. Thompson and Croft 1989, Sanderson 1990 and Stein 1991). This technique presents the results of the query within a single window. Each matched node being represented by short summary information. In query-only retrieval engines this provides a useful, if modest, improvement over scanning of an implicit matched node list. In a system which provides browsing facilities, as well as querying, this technique may be invaluable in making the structure of the document base more explicit. The list of matched nodes would form a *retrieval node*, from which links would emanate to the actual nodes, and in turn to other nodes in the document base. An equivalent structure to that given in figure 7.30 is shown in figure 7.31.

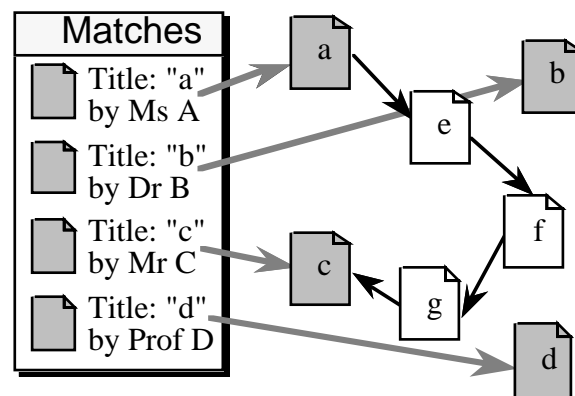


Figure 7.31: Sample network with explicit matched node list

This approach would remove the problem of users not properly scanning down the matched node list, whilst making the structure of the document base more explicit. This approach also leaves the document base unaltered since the extra links to nodes, to connect matched nodes to their successors and predecessors, are no longer required. This approach also concentrates the user's browsing on the matched document list, as this is likely to be where the user will return when a particular path proves fruitless.

This discussion has, so far, assumed that the system can display at least two nodes at a time (the retrieval node and a document content node). If this is not the case then a command would be required to take the user to the retrieval node. This may be considered as providing an extra link from every node to the retrieval node. The benefits of this approach may be slightly reduced in interfaces which only allow one window to open at a time, as the encouragement of returning to the retrieval node may be reduced.

There are two major problems which can be envisaged occurring from the use of retrieval nodes. Firstly, users may miss relevant nodes because their summary information is irrelevant. And secondly, that the provision of suitable summary information in a multimedia document base may be difficult. Al-Hawamdeh *et al.* (1991) showed that reduced views of colour images provided much better summaries than traditionally found in textual systems, but there are still many media for which summary information cannot easily be created.

7.8.2 Presentation of Matching Score

Whenever demonstrating *mmIR*, the issue of what the matching score represented was continuously raised⁵. During the user tests, users did not appear to make use of the score (or matching weight information). The score of matched documents appears to be a major potential source of information for the user, which was not presented well enough in *mmIR*. There are two methods which may be used to display the score of a single node: simply display the score as calculated by the retrieval engine (direct scoring), or rank the first node at 100% and scale the retrieval score of all other nodes accordingly (relative scoring). The second choice provides a more user-oriented view of the matching score. However, it does not provide any evidence on how well the first node performed; for example, it is inconsistent to display nodes with matching values of 0.1 and 0.9 (on a scale from 0 to 1) both as 100% matches. One solution is to present the two variables within a two dimensional graph, for example as used in the Macintosh MedLine system, where the horizontal access represents the matched nodes and the vertical access gives their score. A marker for the currently displayed node is also displayed. As can be seen in figure 7.32 this approach can clearly show the overall retrieval outline as well as the performance of the first match. Plateaus on the graph are also very clear and show areas of the matched node list with similar scores.

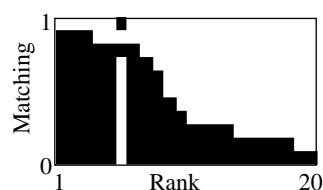


Figure 7.32: Graph of score against position in matched node list

When using a retrieval system which can display many nodes at once, as may be the case for a system using the hybrid model, this approach becomes rather clumsy as the graph would have to be duplicated for each window or multiple markers used within a single graph. A solution to this problem would be to present the graph as part of the retrieval node in a similar manner to I³R (Thompson and Croft 1989). This would allow

⁵ This issue was raised considerably more by information retrieval / research groups than by the sample users.

the user to easily see plateaus in the retrieval scores and to rate documents relative to one another. Figure 7.33 shows an example of such a header.

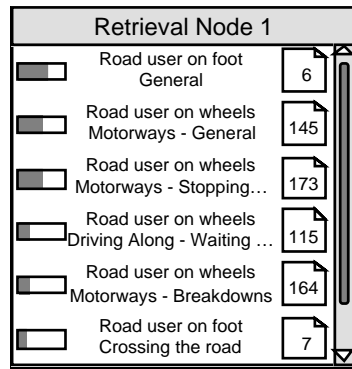


Figure 7.33 Example retrieval node with score graph

The area given to displaying the scores may prove too small to adequately represent the score. It may be desirable to present the scores relative to the first matched query. In this case the absolute score of the best matched node may be presented separately within the retrieval node. The absolute score of the best match could be presented as an estimate of the quality of the retrieval, and as such, may be presented by various textual bands rather than a less understandable percentage. Example bands could be: very poor, poor, reasonable, good, very good, and excellent. The textual bands also allow the retrieval engine to give meaningful feedback on the performance of the system over various document bases, e.g. excellent may vary between over 50% and over 95% depending on the document base and absolute scoring method.

7.8.3 Provision of Maps

The provision of maps, to help users understand their position in the document base, may be severely affected by the existence of a matched node list. This list will typically contain nodes from many different locations, within the hypermedia network, resulting in a wider spread of links than commonly found in hypermedia document bases, and in a reduction of the help provided by maps. If the users' searches were based around an explicit matched document list (e.g. by the provision of retrieval nodes), the requirement for maps, and navigation aids in general, would diminish. The nodes in the document base which are relevant to the user should not be very distant from the matched document list. Thus the chances of users getting lost are reduced because they should never have to browse far from this list. If they do get lost they can also easily return to the list and resume their search from there. Akscyn, McCracken, and Yoder (1988) suggest that, for hypermedia systems, the provision of fast node access and a fast history command significantly reduces the navigation problems, and almost eliminates the requirement for other navigation aids. This suggestion may be even more applicable when considering access via the hybrid of querying and browsing suggested here.

7.8.4 Interface Style

The language used in the interface of a retrieval engine must be very carefully considered and worded in terms of user concepts. The prototype application contained several error messages which tended to confuse users, purely because of their wording. The most common confusion of this nature occurred when the user clicked the go-better command while viewing the best match. An error message was displayed stating “there is no more relevant document than the current one.” Although this is true when considering the retrieval engine’s model of relevance, it is obviously not true from the user’s viewpoint, when the best match is irrelevant and there are relevant documents. A better error message would have been, for example, “you are already viewing the highest scoring document”. In this case it may actually have been better simply to prevent the user from issuing this command by disabling the button. This consideration does, however, raise one of the classic visual interface dilemmas: should users be prevented from doing inappropriate operations, or should they be allowed to do them, so that information can be given on why they are inappropriate? In practice the merits of giving addition information must be weighed against the inconvenience of the user processing the error message on a operation by operation basis. A user interface to a system, which is expected to be used by novices/casual users regularly, must be designed to reduce the number of error causing paths which are open to the user. Provision of good (and fast) help facilities should be used to provide the bulk of information on how to use the system – not error messages.

7.9 Limitations of Tests

Although the tests described in this section have shown many features of access to the chosen document base it would be interesting to carry out further tests in this area. There are four main areas by which these tests could be supplemented. Firstly, the testing of a set of users using the paper based Highway Code would provide a useful base case to compare the on-line versions with. Initial, un-reported, and informal tests using paper based access showed that users found the tasks much harder than they expected and took a reasonable period of time to answer questions.

It would be useful to simply repeat the tests described here so that the sample set of users was increased, in the hope that this would provide more stable results. The tests could also be re-organised to bring users to a reasonable quality, by use of tutorials, before starting the tests. This would give a better feel for how well users will perform with each access method when they know the system well.

Finally, it would be a a better approximation to the real use of these models if the tests were repeated using a much larger document base with a higher ratio of textual to non-textual nodes.

7.10 Conclusions

The user tests which were presented in this section did not show any access method was significantly better, or worse, than any other access method overall. Browsing-only users were significantly faster for the first session than other users, with query and hybrid users taking approximately the same time. During the second session query and browsing access were very similar in speed, with hybrid access trailing slightly. The benefit for browsing access during the first session is as expected. The general belief is that browsing access provides a benefit for novice users. Query-only users performed fastest for hard questions, also as expected. Hybrid access did perform slightly poorer than other access methods overall, but the difference could easily be overcome with a more integrated user interface. The benefits gained from such an interface should be more significant than in an equally improved browse-only or query-only system.

The users' scores did not vary significantly between each access method. From a consideration of the effectiveness of each access method, these tests did not show any difference. There was, however, a slight difference in how well users perceived they had performed. Query-only users appeared to have slightly less confidence in their solutions than other users, while browsing-only and hybrid users had very similar confidence in their answers.

The most obvious reason for hybrid-based access being slower than query-based access, for any given query, would be the relative simplicity of query-based access. With query-only access users simply issue a query and then scan down the matched list. Whereas hybrid access users may also browse around the neighbourhood. With the provision of header nodes for queries, this problem may be significantly reduced, as users will be able to select promising areas of the document base from the retrieval node's matched document list, and then browse through these areas. The provision of retrieval nodes would also reduce the disorientation problems facing users, and should result in an access method which very rarely performs poorer than query-only access.

A run through of one test for each access method by the author was also reported. This test showed the best realistic performance which could be expected of users. When compared with the actual users' results this showed that no access method was closer to the fastest, realistic, access time. The only area in which the expert user performed differently between each access method was in the number of nodes visited. This did not affect the time taken, and thus, the overall performance of the user.

The tests also showed that the users of browsing-only access had slightly more confidence in their solutions than other users. This is likely to be a result of them seeing more of the document base, as they browse, and thus having more evidence to base their confidence on. For each access method users were evenly split between those who felt they could answer the questions better with the computer based Highway Code or a paper based Code. Considering the reasonably small document base and the interface reservations which have been expressed, this is an encouraging figure which shows that

users are willing to use computer-based document bases – even when it would be feasible to search paper based versions.

The user tests did not give any evidence that hybrid access was any poorer than other access methods, and considering other factors and with the suggested improvements to the user interface, the method would provide a simple and efficient access method to mixed media document bases.

The analysis of query styles showed that users did tend to issues reasonably long queries which were often full natural language sentences, although when asked to describe the matching algorithm they typically described it as some form of term-based retrieval. Users were also quite happy to provide relevance feedback, although many users did not understand how to use the feedback once given and very few users correctly used negative feedback.

7.11 Sample Questionnaires

This chapter appendix presents some sample questionnaires, the first questionnaire (four pages in length) gives an example of the entire form which a user would be given during the first session. This questionnaire is composed of an introductory page, an evaluation of the users experience, the test itself, and an evaluation of how the user felt (s)he performed. The second questionnaire (again four pages) shows what the same user would be presented with at the second session, this is very similar to the first except that there is no experience test and there is an additional test to assess how the users understood the system. The example forms presented here are based on the user using the query only version of the interface. Additional pages are also given which show the understanding checks for the browsing-only and hybrid interfaces. Only the Highway Code question pages and the final understanding check are changed between different access models.

At the end of this chapter the three guide sheets are reproduced. Each user was given the appropriate guide sheet as there only information on how to access the document base.

It should be noted that the style format of the alternative questionnaires has been very slightly altered to conform to the page specifications required of a thesis – the general layout is, however, the same as that presented to users. The query-only questionnaires and guide sheets are reproduced at approximately 75% of the original size to fit within thesis page specifications.

8 Conclusions

This thesis has presented much work in the area of retrieving documents held in a document base which provides access by query and by browsing (a hybrid document base). Initially a review of information retrieval was given. This review discussed document retrieval (query-based access), hypermedia (browsing-based access), the combination of both, and methods of accessing non-textual documents. The review also defined, using set notation, the structure of document bases required to provide access by querying, browsing, and their combination. Formal definitions were also given for the document bases required to support access method, the main algorithms used within the access methods, and of evaluation techniques (including a proposal for a complexity definition for hypermedia document bases).

A general model of access to non-textual documents held within a hybrid document base was developed in chapter 3. The basic model provides access to non-textual nodes based on their context in the document base, while accessing textual nodes by content. This model treats a non-textual node, for retrieval purposes, as being the *average* of the documents which are linked with it. This model is then extended to describe all documents partly in terms of their content, if this is possible, and partly in terms of their context in the document base. A general definition is given for a recursive method of defining a document in terms of its content and context. This general algorithm is then specialised to define the context only in terms of the document's immediate neighbours, and then in terms of its immediate neighbours and their immediate neighbours. The model was further extended to support more complex linking strategies (e.g. typed links) and to account for the mixed-media queries (e.g. natural language and impression-based queries). Overall the variations of the model which are developed provide a suite of methods to provide access to non-textual nodes by query, and which define other nodes partly in terms of their context.

To test the hypothesis that the model developed in chapter 3 provides a worthwhile method of accessing non-textual nodes an experiment was run using the text-only CACM collection. The use of a text-only environment, although different from the environment in which the model will be used, provides a method for quantitatively assessing the model. Each record in the CACM collection, which was either cited by or cited another record, was described by two methods. Firstly, the document was described in terms of its content, using standard term based indexing techniques. Secondly, the document was treated as if it were non-textual and a description was calculated using its context (as defined by citations). These descriptions were then compared, using a standard document matching algorithm, to assess how similar the different descriptions were, and consequently, how well the model of non-textual retrieval performed. To provide a base case, the experiment was repeated using randomly created links, rather than citations, to define a document's context. The experiments showed that, when using citations, the

model developed in chapter 3 achieves a similarity of 23% with the index based description. This compares favourably with a similarity of only 4% when the context is defined by random links (approximately $\frac{1}{6}$ th that of citations). Although the similarity is still quite low for citation based links, the experiment did show that citation-based contexts produced significantly better descriptions than randomly created contexts. The low figure for citation-based context is, partly, due to the simplistic information retrieval engine which was used in the experiment. The retrieval engine used basic retrieval technology and, for example, did not use a thesaurus when defining how similar two document descriptions were. The model of describing documents in terms of their context was also the simpler of the two main models developed in chapter 3. Improving on these factors would help to increase the matching between citation-based descriptions and index-based descriptions. The improvement would, however, either not be apparent or would be very small for random based links. Overall the experiment presented in chapter 4 shows that, when links are meaningful, descriptions of documents based entirely on their context are reasonable and would provide a useful method for accessing non-textual documents.

The combination of browsing and querying within the same retrieval system has many consequences, one of which is the effect on relevance feedback. In query only systems relevance feedback, by which users state that a particular document is relevant or irrelevant, can only be given on documents which matched a query – since these are the only documents the user can view. However, in systems which can be accessed by browsing and querying the user can view, and hence give feedback on, non-matching documents. A model was developed, in chapter 5, for correlating the effect that giving feedback on a document has on the query, with how well the document matched the last query. This model was based on the vector space model. While the results shown for positive feedback would be expected to translate to other retrieval model, the effects for negative feedback are not. The model, with experimental support, showed that effect of feedback is not constant with respect to the strength of the original matching. Further it shows that, under the vector space model, the effect of giving positive feedback decreases as the strength of the original matching increases. This is as expected, since giving positive feedback on a non-matching node shows a considerable difference between the user's model and retrieval engine's model of relevance. Whereas, giving positive feedback on a perfect match simply reinforces the retrieval decision. Unfortunately, however, the effects of negative feedback, under the vector space model, are not as expected. As the strength of the original matching increases the effect of negative-feedback decreases, as expected, but at a certain value of original matching the effect starts to increase. This increase is such that giving negative feedback on a perfect match has no effect, despite this showing a major difference between the user's model and the system's model of relevance. Overall the effects of negative feedback were also stronger than for positive feedback. Together with the shape of the relationship between original matching and strength of feedback, this shows that negative feedback cannot be considered as an inverse operation to positive

feedback. The process of providing positive feedback on a document, issuing a relevance-based query, providing negative feedback on the same document, and finally performing another relevance-based query does not, in general (under the vector space model), return the user to the initial state. Together with results from chapter 7 which show users did not understand negative feedback, this raises serious implications on the usefulness of negative feedback.

To test the model of retrieval, developed in chapter 3, in practice and to assess various user interface issues, a prototype applications was developed – *mmIR*. Chapter 6 described the users view of *mmIR* and many of the major implementation details. *mmIR* provides access to the British Highway Code by browsing, querying, or a combination of both and uses the simple model of context-based access to non-textual nodes which was developed in chapter 3. This development work, as well as being essential for user testing, showed that a retrieval system could be built on these models of access and that the system could run effectively on a, reasonably powered, personal computer.

User tests were carried out using *mmIR* as a test retrieval system. Thirty users were used to carry out tests which ran for, approximately, 1 hour, followed by a half hour test, one week later. The users were split so that ten users accessed the document base using only queries, ten using only browsing, and ten using a combination of browsing and querying. The results, presented in chapter 7, showed that, overall, no group of users performed significantly better or poorer than another group. In the early stages of the tests, when users were still learning the basics of the system, users of browsing-only access tended to perform slightly better than other users. At the other end of the difficulty scale, query-only users tended to perform slightly better on the harder questions at the end of each session. Although these results backed up the common belief about query-access compared with browsing-access, the effects were not very large, and for the initial advantage of browsing-access, short lived. Overall, in terms of time taken to answer questions, there was a very small advantage for browsing-based users during the first session, and a slight disadvantage for hybrid-based users in the second session. The mean time taken to answer questions also fell considerably for all access methods between the two sessions. When considering the quality of results, the major criteria when there is little difference in speed, there were no significant differences between the access methods. Query-based users did, however, tend to have slightly less confidence in their answer than other users – this is possibly due to the very restricted view they have of the document base.

Overall, the users tests did not show that hybrid access provided a less effective or slower access method than browsing-only or query-only access. Taking into account the potential benefits within a large document base and the limitations of the user interface, as described in chapter 7, providing access to a hybrid document base by query and by browsing would appear to be a natural approach which is no worse than other access methods despite the potential added complexity.

At the end of each session users of each access method were presented with a questionnaire to check their understanding of the system. Amongst revealing other features, these forms exposed an interesting difference between how users describe the matching algorithm (for queries) and how they formulate their queries. Despite the vast majority of queries being natural language sentences, in many case to the point of correct punctuation, when asked how documents were matched 89% of users described the process as using term based techniques. The user logs also showed that users' queries tended to become longer, for query-only access, as tests progressed. These results appear to show that, although they realise that the system is not performing natural language analysis, users find natural language queries very natural and effective.

In summary: this thesis has shown that access can be made to a document base of mixed media by using techniques from clustering research. The provision of a mixed-media hypermedia system with query facilities is possible. The thesis has also shown that this approach provides a powerful method of accessing information which, with a good user interface, will provide the benefits of both hypermedia browsing and free text querying without introducing an overly complex access model.

9 Bibliography

- Adams, D., “Hyperland”, *BBC Television (BBC2 TV Programme)*, London, September 1990
- Aigrain, P., and Longueville, V., “A connection graph for user navigation in a large image bank”, proceedings of the RIAO 91 conference on *Intelligent Text and Image Handling*, Universitat Autònoma de Barcelona, Catalunya, Spain, pp. 67–84. (April 1991)
- Akscyn, R.M. , McCracken, D.L., and Yoder, E.A., “KMS: A distributed hypermedia system for managing knowledge in organizations”, *Communications of the A.C.M.*, vol **31** (7), pp. 820-835. (July 1988)
- Al-Hawamdeh, S., Ang, Y.H., Hui, L., Ooi, B.C., Price, R., and Tng, T.H., “Nearest neighbour searching in a picture archive system”, proceedings of ACM SIGIR / ISS *International Conference of Multimedia Information Systems*, Institute of Systems Science, National University of Singapore, pp. 17-34. (January 1991(a))
- Al-Hawamdeh, S., and Ooi, B.C., “Semantic-based query formulation in PAS”, proceedings of the RIAO 91 conference on *Intelligent Text and Image Handling*, Universitat Autònoma de Barcelona, Catalunya, Spain, pp. 916-933. (April 1991(b))
- Apple Computer Inc., *HyperCard™ User’s Guide*, Apple Computer, Cupertino, California, U.S.A. (1987)
- Begeman, M.L. and Conklin, J., “The right tool for the job”, *Byte*, vol.**13**(10), pp. 255-266. (October 1988)
- Begoray, J.A. , “An introduction to hypermedia issues, systems and application areas”, *Int. J. Man-Machine Studies*, vol. **33**, pp. 121-147. (1990)
- Bovey, J.D., and Robertson, S.E., “An algorithm for weighted searching on a boolean system”, *Information Technology: Research and Development*, vol **3**(2), pp. 84–87. (April 1984)
- Bovey, J.D., and Brown, P.J., “Interactive document display and its use in information retrieval”, *Journal of Documentation*, vol. **43**(2), pp. 125-137. (June 1987)
- Brown, P.J., “Interactive documentation”, *Software: Practice and Experience*, vol. **16**(3), pp. 291–299. (March 1986)
- Burkowski, F.J., “The use of retrieval filters to localize information in a hierarchically tagged text-dominated database”, proceedings of the RIAO 91 conference on *Intelligent Text and Image Handling*, Universitat Autònoma de Barcelona, Catalunya, Spain, pp. 264-284. (April 1991)
- Bush, V. , “As we may think”, *Atlantic Monthly*, pp. 101-108. (July 1945)

- Clemencin, G., “Querying the French ‘Yellow Pages’: Natural language access to the directory”, *Information Processing and Management*, vol. **24**(6), pp. 633-649. (1988)
- Cleverdon, C.W., Mills, L., and Keen, M., *Factors Determining the Performance of Indexing Systems*, ASLIB, Cranfield Project, Cranfield. (1966)
- Conklin, J., “Hypertext: an introduction and survey”, *IEEE Computer*, vol. **20**(9), pp. 17-41. (September 1987)
- Constantopoulos, P., Drakopoulos, J., and Yeorgaroudakis, Y., “Retrieval of multimedia documents by pictorial content: a prototype system”, proceedings of ACM SIGIR / ISS *International Conference of Multimedia Information Systems*, Institute of Systems Science, National University of Singapore, pp. 35-48. (January 1991)
- Croft, W.B., *Organizing and searching large files of documents*, Ph.D. Thesis, Churchill College, Cambridge. (1978)
- Croft, W.B., and Harper, D.J., “Using probabilistic models of document retrieval without relevance feedback”, *Journal of Documentation*, volume 35, pp. 285-295. (1979)
- Department of Transport, *The Highway Code*, Her Majesty’s Stationary Office, London, U.K. (1988)
- Ducloy, J., Grivel, L., Lamirel, J., Polanco, X., and Schmitt, L., “INIT’s experience in hyper-document building from bibliographic databases”, proceedings of the RIAO 91 conference on *Intelligent Text and Image Handling*, Universitat Autònoma de Barcelona, Catalunya, Spain, pp. 25–44. (April 1991)
- Englebart, D.C., and English, W.K., “A research center for augmenting human intellect”, *AFIPS Conference Proceedings*, vol. **33**(1), pp. 395–410. (December 1968)
- Enser, P.G.B., “An index-free approach to the retrieval of still images”, presented at the British Computer Society 13th Information Retrieval Colloquium, University of Lancaster. Paper available from Department of Information and Library Studies, University College of Wales, Aberystwyth, U.K. (April 1991)
- Fox, E.A., and Koll, M.B., “Practical enhanced boolean retrieval experiences with the SMART and SIRE systems”, *Information Processing and Management*, vol. **24**(3), pp. 257–267. (1988)
- Frei, H.P., and Jauslin, J.F., “Graphical presentation of information and services: a user-oriented interface”, *Information Technology: Research and Development*, vol. **2**, pp. 23–42. (1983)
- Frisse, M.E., “Searching for information in a medical handbook”, *Communications of the ACM*, vol. **31**(7), pp. 880-886. (July 1988)

- Frisse, M.E., “Information retrieval from hypertext: update on the Dynamic Medical Handbook project”, proceedings of the second ACM conference on hypertext, *Hypertext '89*, Pittsburgh, U.S.A., pp. 199-212. (November 1989)
- Furuta, R., and Stotts, P.D., “Programmable browsing semantics in Trellis”, proceedings of the second ACM conference on hypertext, *Hypertext '89*, Pittsburgh, U.S.A., pp. 27-42. (November 1989)
- Garg, P.K., “Abstraction mechanisms in hypertext”, *Communications of the ACM*, vol. **31**(7), pp. 862-870,879. (July 1988)
- Halasz, F.G., “Reflections on NoteCards: seven issues for the next generation of hypermedia system”, *Communications of the ACM*, vol **31** (7), pp. 836-852. (July 1988)
- Halin, G., Créhange, M., and Kerekes, P., “Machine learning and vectorial matching for an image retrieval model: EXPRIM and the system VISAGE”, proceedings of the 13th ACM SIGIR conference, Brussels, Belgium, pp. 99–114. (September 1990)
- Harabayshi, F., Matoba, H., and Kasahara, Y., “Information retrieval using impression of documents as a clue”, proceedings of 11th ACM SIGIR conference, Grenoble, France. (June 1988)
- Heine, M.H., “A logic assistant for the database searcher”, *Information Processing and Management*, vol. **24**(3), pp. 323–329. (1988)
- Hofmann, M., Langendörfer, H., Laue, K., and Lübben, E., “The principle of locality used for hypertext presentation: navigation and browsing in CONCORDE”, proceedings of the British Computer Society HCI '91 conference *People and Computers VI*, Heriot-Watt University, Edinburgh, U.K., pp. 435–452. (August 1991)
- Humphries, J., *Highway Code questions and answers*, Elliot Right Way Books, Kingswood, Surrey, U.K. (1987)
- Ide, E., “Relevance feedback in an automatic document retrieval system”, M.Sc. thesis, Report ISR-15, National Science Foundation, Department of Computer Science, Cornell University, N.Y., U.S.A. (1969)
- Kato, T., Kurita, T., Shimogaki, H., Mizutori, T, and Fujimura, K., “A cognitive approach to visual interaction”, proceedings of ACM SIGIR / ISS *International Conference of Multimedia Information Systems*, Institute of Systems Science, National University of Singapore, pp. 109-120. (January 1991)
- Kurlander, D., and Bier, E.A., “Graphical search and replace”, *ACM Transactions on Computer Graphics*, vol. **22**(4), pp. 113-120. (August 1988)

- Morrissey, J.M., Harper, D.J., and van Rijsbergen, C.J., “Interactive Querying Techniques for an Office Filing System”, *Information Processing & Management*, vol. **22**(2), pp. 121-134. (1986)
- Neilsen, J., *Hypertext and Hypermedia*, Academic Press, San Diego, California. (1990)
- Nelson, T.H., “Getting it out of our system”, *Information Retrieval: A Critical Review*, G. Schechter, ed., Thompson Books, Washington D.C. (1967)
- Nelson, T.H., *Literary Machines*, Mindful Press, Sausalito, California. (1990)
- Oddy, R.N., “Information Retrieval Through Man-Machine Dialogue”, *The Journal of Documentation*, vol. **33**(1), pp. 1-14. (March 1977)
- Pausch, R. and Detmer, J., “Node popularity as a hypertext browsing aid”, *Electronic Publishing Origination, Dissemination and Design*, vol. **3**(4), pp. 227–234. (November 1990)
- Porter, M.F. , “An Algorithm for Suffix Stripping”, *Program*, vol. **14**(3), pp. 130-137. (July 1980)
- Rabitti, F., and Savino, P., “Automatic image indexation and retrieval”, proceedings of the RIAO 91 conference on *Intelligent Text and Image Handling*, Universitat Autònoma de Barcelona, Catalunya, Spain, pp. 864-884. (April 1991)
- Raymond, D.R., and Tompa, F.W., “Hypertext and the Oxford English Dictionary”, *Communications of the ACM*, vol. **31**(7), pp. 871-879. (July 1988)
- Salton, G. (editor), *The SMART Retrieval System*, Prentice Hall, New Jersey. (1971)
- Salton, G., “A simple blueprint for automatic boolean query processing”, *Information Processing and Management*, vol. **24**(3), pp. 269–280. (1988)
- Sanderson, M., and Van Rijsbergen “NRT (News Retrieval Tool)”, *Electronic Publishing Origination, Dissemination and Design*. (to be published 1992)
- Savoy, J, and Desbois, D., “Baysian inference networks in hypertext”, proceedings of the RIAO 91 conference on *Intelligent Text and Image Handling*, Universitat Autònoma de Barcelona, Catalunya, Spain, pp. 662-681. (April 1991)
- Smeaton, A.F., and Sheridan, P. “Using morpho-syntactic language analysis in phrase matching”, proceedings of the RIAO 91 conference on *Intelligent Text and Image Handling*, Universitat Autònoma de Barcelona, Catalunya, Spain, pp. 414–430. (April 1991)
- Sparck Jones, K., *Automatic Keyword Classification in Information Retrieval*, Butterworths, London. (1971)
- Sparck Jones, K., “A statistical interpretation of term specificity and its application in retrieval”, *Journal of Documentation*, **28**, pp. 11-21. (1972)

- Sparck Jones, K. and Webster, C.A., “Research on relevance weighting”, British Library Research and Development Report 5553, Computer Lab., University of Cambridge. (1980)
- Stein, R.M., “Browsing Through Terabytes”, *Byte*, vol. **16**(5). (May 1991)
- Stotts, P.D., and Furuta, R., “Petri-Net-Based Hypertext: Document Structure with Browsing Semantics”, *ACM Transactions on Information Systems*, vol. **17**(1), pp. 3-29. (January 1989)
- Symantec, *Think’s Lightspeed Pascal User’s Manual*, Symantec, Cupertino, U.S.A. (1988)
- Tague, J., and Schultz, R., “Some measures and procedures for evaluation of the user interface in an information retrieval system”, proceedings of the 11th ACM SIGIR conference, Grenoble, France, pp. 371-385. (June 1988)
- Thompson, R.H., and Croft, W.B., “Support for browsing in an intelligent text retrieval system”, *International Journal of Man-Machine Studies*, vol. **30**(6). (June 1989)
- Tompa, F.W., “A data model for flexible hypertext database systems”, *ACM Transactions on Information Systems*, vol. **7**(1), pp. 85-100. (January 1989)
- van Rijsbergen, C.J., *Information Retrieval (second edition)*, Butterworths, London. (1979)
- Verhoeff, J., Goffman, W., and Belzer, J., “Inefficiency of the use of boolean functions for information retrieval systems”, *Communications of the ACM*, vol. **4**(12), pp. 557-558, 594. (December 1961)
- Waltz, D.L., “Applications of the Connection Machine”, *IEEE Computer*, vol. **20**(1), pp. 85–97. (January 1987)
- Watters, C., and Shepherd, M.A., “Transient hypergraph-graph based model for data access”, *ACM Transactions on Information Systems*, vol. **8**(2), pp. 77-102. (April 1990)
- Williams, M.D., “What Makes Rabbit Run?”, *International Journal of Man-Machine Studies*, vol. **21**, pp. 333-352. (1984)
- Wilson, E., “Integrated information retrieval for law in a hypertext environment”, proceedings of the 11th ACM SIGIR conference, Grenoble, France, pp. 663-677. (June 1988)
- Yankelovich, N., and Meyrowitz, N., “Reading and writing the electronic book”, *IEEE Computer*, vol. **18**(10), pp. 15-29. (October 1985)
- Zernik, U., “Train vs. Train: Tagging Word Senses in Corpus”, proceedings of the RIAO 91 conference on *Intelligent Text and Image Handling*, Universitat Autònoma de Barcelona, Catalunya, Spain, pp. 567–585. (April 1991)